Tampereen yliopisto

Tuomas Lindroos

TIEDONSIIRTO SULAUTETUSTA JÄRJESTELMÄSTÄ NB-IOT VERKOSSA

Tieto- ja sähkötekniikan tiedekunta
Diplomityö
Kesäkuu 2019

TIIVISTELMÄ

TUOMAS LINDROOS: Tiedonsiirto sulautetusta järjestelmästä NB-IoT-verkossa
Tampereen teknillinen yliopisto
Diplomityö, 61 sivua, 1 liitesivu
Kesäkuu 2019
Sähkötekniikan maisterin tutkinto-ohjelma
Pääaine: Sulautetut Järjestelmät
Tarkastajat: professori Karri Palovuori ja professori Jukka Vanhala

Avainsanat: Esineiden Internet, Teollinen Internet, Mbed-OS, NB-IoT, LPWAN, LwM2M, CoAP, BLE

Kasvava verkkoon kytkettävien antureiden määrä on aiheuttanut tarpeen uusille laajan kantaman vähävirtaisille langattomille tietoliikennetekniikoille. Tässä työssä keskitytään tutkimaan yhden tällaiseen tarpeeseen vastaavan verkon mahdollisuuksia. Tekniikan nimi on Narrow Band Internet of Things (NB-IoT). Sen kokeilemiseksi työssä toteutettiin sulautettu järjestelmä, joka kommunikoi kyseisen verkon kautta käyttäen kevyttä lwM2M-tiedonsiirtoprotokollaa.

Sulautetulle laitteelle, joka toimi lwM2M-protokollan asikaslaitteena, toteutettiin myös saman protokollan palvelinosapuoli, jonka kanssa se voi kommunikoida. Kokonaisjärjestelmän mielekkyyden vuoksi sillä haluttiin kerätä dataa myös oikeilta antureilta ja lähettää se pilvipalvelimelle. Datan lähteiksi valkoitui joukko Bluethooth Low Energy (BLE) -laitteita, joiden yhteyspisteenä työssä rakennettu laitteisto toimii. Lopputuloksena saatiin siis dataputki, jossa BLE-laitteelta lähtöisin oleva data siirtyy NB-IoT-verkon kautta lwM2M-protokollaa käyttäen asiakkaalta palvelimelle ja lopulta HTTP-yhteyden kautta lopulliselle pilvipalvelimelle, jossa se säilötään tietokantaan. LwM2M-protokolla tarjoaa myös tiedonsiirron palvelimelta asiakkaalle mahdollistaen yhteyspisteen hallitsemisen etänä.

ABSTRACT

TUOMAS LINDROOS: Data Transmission from an Embedded Device in the NB-IoT Network
Tampere University of Technology
Master of Science Thesis, 61 pages, 1 Appendix page
June 2019
Master's Degree Programme in Electrical Engineering
Major: Embedded Systems
Examiners: Professor Karri Palovuori and Professor Jukka Vanhala

Keywords: Internet of Things, Mbed-OS, NB-IoT, LPWAN, LwM2M, CoAP, BLE

As there is constantly growing number of sensors connected to internet, there is a need for new low power network technologies that cover wide area. In this thesis the main focus is to investigate the possibilities of one such network. This technology is called Narrow Band-Internet of Things (NB-IoT). To try out this new technology an embedded system was implemented, that communicates via NB-IoT using Lightweight Machine to Machine (LwM2M) communication protocol.

For the embedded system that worked as client in LwM2M protocol to communicate, also an LwM2M server was implemented. Also for the entire system to make sense, there was a need to send actual sensor data to cloud server. a Group of Bluetooth Low Energy (BLE) devices was chosen as an origin of the sensor data, and the embedded device acts as gateway for the BLE devices. As an end result a complete data pipe was completed where data originating from BLE devices is transferred via NB-IoT network from LwM2M client to server and is forwarded with HTTP protocol to it's final destination in cloud server. LwM2M protocol makes it also possible to send messages from server to client. This feature is used to configure the gateway over the air.

ALKUSANAT

Tämä diplomityö on laadittu Wapice Oy:n tarjoamasta aiheesta maaliskuun 2018 ja kesäkuun 2019 välisenä aikana. Työn ohjaajana on toiminut Diplomi-Insinööri Mickey Shroff. Haluan kiittää häntä ja muita Wapice Oy:n työntekiöitä, jotka ovat kannustaneet ja myötävaikuttaneet hyvään työskentelyilmapiiriin.

Haluan esittää myös erityisen kiitoksen Tampereen teknillisen yliopiston puolesta tarkastajana toimineille professori Karri palovuorelle ja professori Jukka vanhalalle. Suuren kiitoksen esitän myös tyttöystävälleni, josta on ollut suuri apu työn oikoluvussa.

Lopuksi haluan kiittää perhettäni ja ystäviäni hyvästä kannustuksesta, jota olen heiltä saanut opiskelujeni aikana.

Tampereella, 10.06.2019

Tuomas Lindroos

SISÄLLYSLUETTELO

KUVALUETTELO

LYHENTEET JA MERKINNÄT

IoT	engl. *Internet of Things*, Esineiden internet
NB-IoT	engl. *NarrowBand IoT*, Verkkoteknologia esineiden internetille
M2M	engl. *Machine to Machine*, Laitteiden välinen kommunikointi
LWM2M	engl. *Light Weight M2M*, Kevyt kommunikaatioprotokolla esineiden internetille
NFC	engl. *Near Field Communication*, matkapuhelimissa käytettävä langaton lyhyen kantaman verkkotekniikka
RFID	engl. *Radio Frequency Identification*, langaton esineiden tunnistamiseen käytettävä radiotekniikka
WiFI	Langattoman lähiverkon kauppanimitys
LPWAN	engl. *Low Power Wide Area Network*, Yleisnimitys pieninergisille laajan kantaman radiotekniikoille
LoRa	engl. *Long Range*, erään LPWAN-verkon fyysinen tiedonsiirtokerros
LoRaWAN	engl. *Long Range Wide Area Network*, LPWAN-teknologia, joka sisältää LoRa:n ja ylemmät tiedonsiirtoprotokollat
BLE	engl. *Bluetooth Low Energy*, Laajennus Bluethooth-standardiin pienienergisten laitteiden käytettäväksi
ICT	engl. *Information and Communication Technologies*, Yleisnimitys eri informaatioteknologioista
3G	engl. *third generation*, Kolmannen sukupolven matkapuhelinverkko
LTE	engl. *Long-Term Evolution*, Neljännen sukupolven matkapuhelinverkkotekniikka
LTE Cat-M1	toinen LTE-verkon LPWAN-laajennuksista NB-IoT:n lisäksi
MAC	engl. *Medium access control*, Verkon varaamisen ja itse liikennöinnin hoitava osajärjestelmä
3GPP	engl. *Third Generation Partnership Project*, Yhteistyöprojekti, joka standardisoi mobiiliverkkoja
GSM	engl. *Global System for Mobile Communications*, Toisen sukupolven matkapuhelinverkko
VoLTE	engl. *voice over LTE*, 4G-verkkojen ominaisuus, jossa äänipuhelut kulkevat 4G-verkon välityksellä
QPSK	engl. *Quadrature Phase Shift Keying*, Nelivaiheinen modulointitekniikka, jossa viestin arvo kerrotaan siirtämällä kantoaallon vaihetta
FDMA	engl. *Frequency Division Multiple Access*, Taajuuskanavointi on tekniikka, jossa laitteet voivat varata kanavia lähettämisen ajaksi
OFDMA	engl. *Ort-hogonal FDMA*, Useamman laitteen versio FDMA:sta
QoS	engl. *Quality of service*, Tarkoittaa tietoliikenteen priorisointia, jossa osaa liikenteestä voidaan hidaastaa, mikäli linjojen tiedonsiirtokapasiteetti ei riitä.
HTTP	engl. *Hypertext Transfer Protocol*, Datansiirtoprotokolla, jota verkkoselaimet ja palvelimet käyttävät tiedonsiirtoon
REST	engl. *Represantional State Transfer*, Arkkitehtuurimalli verkkorajapintojen toteuttamiseen
XML	engl. *Extensible Markup Language*, merkintäkielien yleiskäsite, jossa tiedon merkitys voidaan kirjoittaa tiedon sekaan
HTML	engl. *Hypertext Markup Language*, Kuvauskieli, jolla voi kirjoittaa verkkosivuja

URI	engl. *Uniform Resource Identifier*, Merkkijono, jolla kerrotaan tiedon paikka tiedostojärjestelmässä
CoAP	engl. *Constrained Application Protocol*, Tiedonsiirtoprotokolla rajoittuneille laitteille, jossa energiatehokkuus on keskiössä
UDP	engl. *User Datagram Protocol*, Tietoliikenneprotokolla, joka ei vaadi yhteyden muodostamista datan siirtämiseksi
DTLS	engl. *Datagram Transport Layer Security*, UDP-yhteyksissä käytettävä salausmekanismi
SMS	engl. *short message service*, Tekstiviesti
TCP	engl. *Transmission Control Protocol*, Tietoliikenneprotokolla, jossa muodostetaan yhteyksiä tietokoneiden välille
TLS	engl. *Transport Layer Security*, TCP-yhteyksissä käytettävä salausmekanismi
TLV	engl. *type-lenght-value*, Formaatti, jolla viesti voidaan jäsentää kenttiin tyyppi, pituus ja arvo
IP	engl. *Internet Protocol*, Verkkokerroksen protokolla, joka huolehtii pakettien pääsystä perille
OMA	engl. *Open Mobile Alliance*, Järjestö, joka standardoi LwM2M-protokollaa
OMNA	engl. *OMA Naming Authority*, OMAn osa, johon voi rekisteröidä LwM2M-resurssien nimiä,
GAP	engl. *Generic Access Profile*, profiili joka määrittää laitteen roolin BLE-protokollassa
GATT	engl. *Generic Attribute Profile*, Määrittää resurssimallin BLE-protokollassa
ATT	engl. *Attribute protocoll*, Määrittää jokaiselle BLE-datan osalle tunnisteen
UUID	engl. *universally unique identifier*, Tunniste joka on ainoa laatuaan
Mbed OS	Armin kehittämä käyttöjärjestelmä sulautetuille laitteille
IDE	engl. *integrated development enviroment*, Kehitysympäristö
HAL	engl. *hardware abstraction layer*, Ohjelmointirajapinta, jolla kätketään laitteisto sovellukselta
SPI	engl. *Smal System Interface*, Tiedonsiirtoväylä
I2C	Tiedonsiirtoväylä
RTOS	engl. *Real Time Operating System*, Reaaliaikainen käyttöjärjestelmä
MB	engl. *Mega byte*, Megatavu, suure joka kuvaa datan määrää
kb/s	engl. *kilo byte / second*, kilotavua sekunnissa, suure joka kuvaa tiedonsiirtonopeutta
SRAM	engl. *Static Random Access Memory*, Staattinen luku ja kirjoitusmuisti
MHz	*Megahertsi*, Suure, jolla kuvataan taajuutta
UART	engl. *Universal Asynchronous Receiver Transmitter*, Sarjaliikenneväylä

1. JOHDANTO

Esineiden internetillä (engl. *Internet of Things, IoT*) tarkoitetaan tänä päivänä valtavaa joukkoa antureita ja laitteita, jotka ovat verkon välityksellä keskenään yhteydessä. On ennustettu, että vuoteen 2020 mennessä internetiin on kytketty yli 50 miljardia laitetta [37], ja laitteiden määrä jatkaa yhä kasvuaan. Langattomat verkkotekniikat ovat suuri tekijä kasvun mahdollistamisessa. Langattomuus on erityisen hyödyllinen sovelluskohteissa, kuten etämittauksissa, älykkäissä sähköverkoissa ja rakennuksissa, paikannuksessa, varashälyttimissä, moottoreiden etäohjauksessa ja monissa muissa älysovelluksissa. Tyypillisessä sovelluskohteessa IoT-laite kerää dataa ympäristöstään jollakin anturilla ja lähettää sitä säännöllisesti läheiselle yhteyspisteelle (engl. *Gateway*), jonka kautta sitä voidaan lähettää pilvipalveluun jatkokäsiteltäväksi. Tyypillinen yhteyspiste voi olla esimerkiksi Wi-Fi-yhteyspiste tai jokin muu internetiin yhteydessä oleva tukiasema.

Erilaiset langattomat verkkotekniikat voidaan ryhmitellä kantaman perusteella useisiin ryhmiin. Lyhimmän kantaman ryhmän muodostavat tällöin esimerkiksi mobiilimaksamisessa käytettävä NFC ja opiskelijakorttien kulunvalvonnassa käytettävä RFID. Näiden teknologioiden kantama vaihtelee käytettävästä taajuusalueesta riippuen muutamasta senttimetristä muutamiin metreihin. Tätä laajempaan kantamaan kykenevät Wi-Fi, Bluetooth ja ZigBee, joiden kantama antennista ja olosuhteista riippuen voi yltää sataan metriin. Vielä pidempiin etäisyyksiin päästään erilaisilla matkapuhelinverkkoteknologioilla. [27]

Perinteisten matkapuhelinverkkojen hinnasta ja korkeasta energinakulutuksesta johtuen niiden rinnalle on viime vuosina alettu kehittää erilaisia pienienergisiä laajan alueen verkkoja (engl. *Low Power Wide Area Network, LPWAN*). Tällaisia teknologioita ovat esimerkiksi LoRa, Sigfox ja Nb-IoT. Teknologioiden etuna on matala energiankulutus, alhainen hinta ja suuri toimintasäde, joten ne tarjoavat ratkaisun sovelluskohteisiin, joissa halutaan asentaa halpoja laitteita mahdollisesti suuri määrä ja luottaa siihen, että ne toimivat vuosia ilman huoltoa samalla energianlähteellä. Näiden tekniikoiden haittana ovat matalat tiedonsiirtonopeudet, joten tekniikat on suunnattu pienten datamäärien keräämiseen eikä esimerkiksi videoiden suoratoistoon. [27]

Teollisuudelle sulautettuja järjestelmiä ja IoT-palveluita tajoava Wapice Oy haluaa olla mukana kehittämässä sovelluksia uusille LPWAN-tekniikoille ja tarjota niitä asiakkailleen. Wapicella on olemassa IoT-Ticket-niminen pilvipalvelu, johon voidaan kerätä dataa erilaisilta IoT-antureilta ja sitten visualisoida ja jatkokäsitellä sitä. Työn aiheeksi sovittiin sen tutkiminen, miten IoT-Ticketiin voitaisiin jatkossa kerätä dataa myös NB-IoT-verkosta ja luoda demonstraatiolaite tätä tarkoitusta varten.

Demonstraatiolaitteeksi sovittiin NB-IoT-verkkoa käyttävä Bluetooth-yhteyspiste. Yhteyspisteen tehtävänä on kerätä dataa sitä ympäröiviltä Bluetooth Low Energy (BLE) -antureilta ja lähettää se eteenpäin NB-IoT-verkon kautta IoT-Ticketiin. Järjestelmään haluttiin saada myös mahdollisuus lähettää viestejä toiseen suuntaan, jotta yhteyspisteen asetuksia olisi tarvittaessa mahdollista muuttaa etänä.

Työ on järjestetty viiteen lukuun. Luvussa 2 käsitellään työn kannalta olennainen teoria, joka käsittää yleiskatsauksen LPWAN-tekniikoihin ja erityisesti NB-IoT:n etuihin ja haittoihin muihin verkkoihin verrattuna. Teorialuvussa käsitellään myös tiedonsiirtoprotokollia, joita työssä käytetään datan siirtoon laitteen ja palvelimen välillä. Lisäksi käydään läpi BLE-protokolla ja laitteen toteuttamisessa käytetty kehitysympäristö Arm Mbed. Luvussa 3 käsitellään sitä, miten järjestelmä luotiin sekä laitteen että palvelimen osalta. Luvussa 4 esitetään olennaisimmat tulokset eli valmis järjestelmä. Lopuksi luvussa 5 kerrataan olennaisimmat asiat. Liitteenä on kaavio järjestelmän arkkitehtuurista.

2. TAUSTA

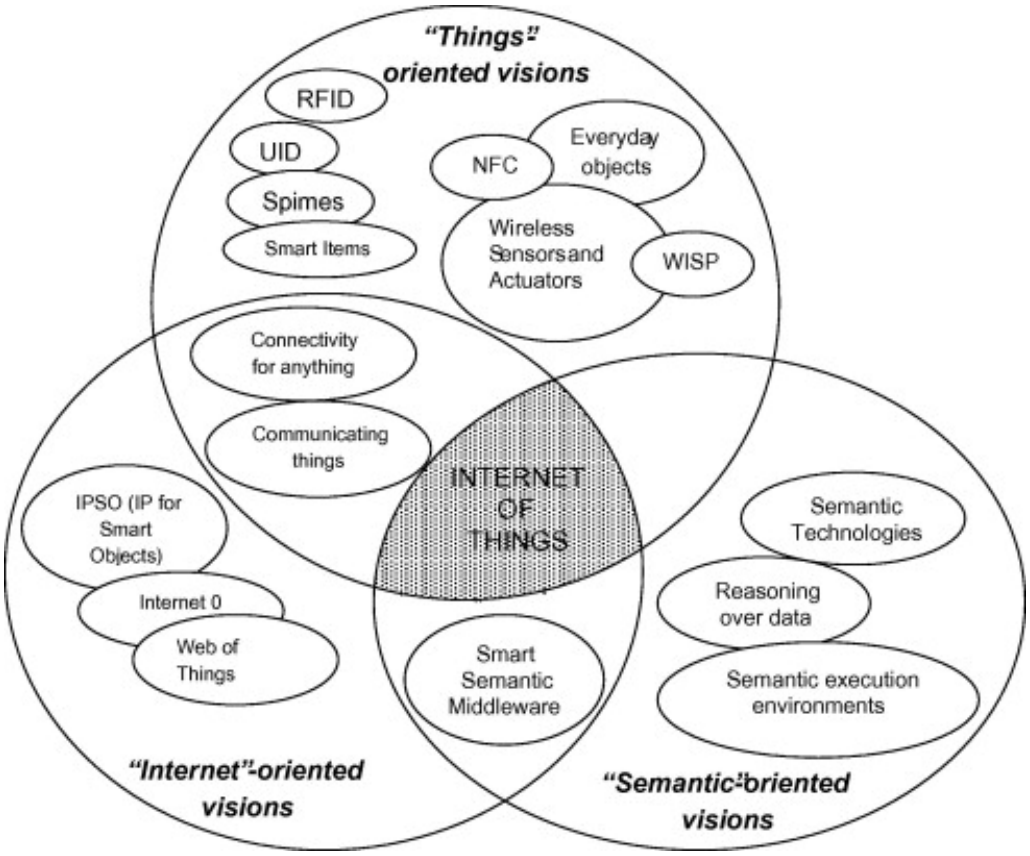
Esineiden internetin toteuttamisen kannalta olennaisimpia teknisiä haasteita ovat verkkoyhteys eli radiotekniikka, jolla informaatio siirtyy kahden pisteen välillä, kommunikointiprotokollat eli datan lähetys muodossa, jonka vastaanottaja ymmärtää, datan salaus, verkkosovelluksen ohjelmointirajapinnat ja datan jälkikäsittely. Erona perinteiseen internetiin on laitteiden valtava määrä ja toisaalta laitteiden rajallinen energiavarasto. Tämä johtaa suunnitteluratkaisuihin, joissa pyritään minimoimaan laitteiden verkkoliikenne ja toisaalta energiaa pyritään säästämään laitteen unitiloilla. Unitilat aiheuttavat osaltaan laitehallintaan ja päivitettävyyteen haasteita, koska laite ei ole jatkuvasti saavutettavissa. Tässä luvussa käsitellään tarkemmin kommunikaatioyhteyttä, protokollia, salausta ja laitehallintaa.

2.1 Esineiden internet

Esineiden internetillä (*Internet of Things*, IoT) tarkoitetaan uutta viitekehystä nykyaikaisessa langattomassa tiedonsiirrossa. Sen keskeisenä ideana on, että erilaiset esineet ympärillämme, kuten mittausanturit ja toimilaitteet, kykenevät kommunikoimaan keskenään ja saavuttamaan yhteisiä päämääriä. IoT koostuu lähteistä riippuen kolmesta suuresta visiosta, joiden erot tulevat tarkastelunäkökulmasta. Näkökulma voi olla joko internetpainotteinen, esinepainotteinen tai semanttinen riippuen esittävän organisaation intresseistä. Eri visiot on esitetty Kuvassa 1. Esineiden internetillä on laaja sovelluskenttä logistiikassa, terveydenhuollossa ja älykkäissä ympäristöissä. Sovelluksia ovat muun muassa avustettu ajaminen, ympäristön tarkkailu, esineiden seuranta, datan keruu ja henkilön tunnistus. [7]

Esineiden internetin yksi keskeinen osa on koneelta koneelle (*machine-to-machine*, M2M) -teknologiat. M2M-teknologioilla viitataan informaatioteknologioihin (*information and communications technologies*, ICT), jotka kykenevät itsenäisesti käsittelemään, mittaamaan ja siirtämään informaatiota sekä reagoimaan siihen. Kyseisissä tekniikoissa ihmisen vuorovaikutustarve konfiguraatio-, toimitus-, käyttö- ja ylläpitovaiheissa pyritään pitämään vähäisenä. Esimerkkejä M2M-tekniikoista ovat kaukomittaukset autojen jarrujen tilasta liikkeessä, vanhusten verenpainemittaukset ja korroosion tarkkailu öljyputkissa. Koneet eivät pärjää ihmiselle luovassa työssä mutta ovat parempia toistuvissa tehtävissä, kuten säännöllisissä mittauksissa ja aikakriittisissä tehtävissä, kuten koneen osien seuraamisessa. [1]

M2M-systeemien datansiirtoa kuvataan englanniksi usein samoin kuin 1900-luvun öljynkäsittelyprosessia termeillä upstream, downstream ja mash-up. [1]



Kuva 1. *Esineiden internetin eri visiot [7]*

- Upstream: Data kerätään kentällä olevilta ilmaisimilta ja siirretään langattomasti yhteyspisteelle suoraan laitteelta tai useamman hypyn kautta *mesh*-verkossa. Data jatkaa yhteyspisteeltä internetissä palvelimelle, johon se tallennetaan käsittelyä varten.
- Downstream: Datan perusteella tehdään päätöksiä, jotka työnnetään takaisin asiakkaalle. Takaisin työnnettävällä datalla voidaan ohjata toimilaitteita, kuten moottoria, tai data voidaan esittää laitteenhallinta-alustalla.
- Mash-up: Kerätty data syötetään tilastollisiin analytiikkatyökaluihin, joiden avulla voidaan tehdä ennusteita ja optimoida prosesseja. Ennusteilla ja optimoinnilla saadaan tehostettua teollisuuden prosesseja.

2.2 Langaton vekkoyhteys

Esineiden internetistä puhuttaessa kommunikoivia antureita ja toimilaitteita halutaan usein asentaa suuri määrä. Tämä johtaa tarpeeseen mahdollisimman helposta ja siten halvasta asentamisesta. Kyseisen tarpeen toteuttamisessa on keskeisessä osassa langaton verkkoyhteys, jota käytettäessä vältytään uusien kaapeleiden asentamiselta. Tilannetta helpottaa edelleen laitteideiden mahdollinen matala energiankulutus, jolloin laitteet eivät tarvitse verkkovirtaa toimiakseen, vaan tulevat toimeen omalla energianlähteellään, kuten paristolla tai aurinkopaneelilla.

Langattomien M2M-verkkotekniikoiden kenttää ovat toistaiseksi hallinneet ZigBee-tyyppiset ja 2G/3G-matkapuhelinverkkoteknologiat (engl. *Cellular network*). Uudempia tulokkaita ovat esimerkiksi matalan tehon Wi-Fi (*low-power Wi-Fi*), matalan energian Bluetooth (*Bluetooth low energy*, BLE) ja patentoidut IoT-verkot. [1]

ZigBee-tyyppisissä tekniikoissa data kulkee solmulta toiselle lyhyinä hyppyinä. Tekniikka on saatu optimoitua erittäin energiatehokkaaksi. Verkolle on kuitenkin luontaista siirtää varsin suuria datamääriä lyhyitä matkoja. Tämä on vastoin M2M-ideaa, jossa tarkoituksena on saada siirrettyä pieniä datamääriä pitkiä matkoja. Verkon laajuuden lisääntyessä se monimutkaistuu, koska viestejä välittävien toistimien määrä lisääntyy, mikä vaatii paljon suunnittelutyötä. Monimutkaisuutensa takia ZigBee ei ole onnistunut muuttumaan asiakkaille myyntäväksi käänteentekeväksi ratkaisuksi. Teknisestä vajavaisuudestaan huolimatta tekniikka on onnistunut pääsemään sertifioiduksi ratkaisuksi langattomissa valvomosysteemeissä. [1]

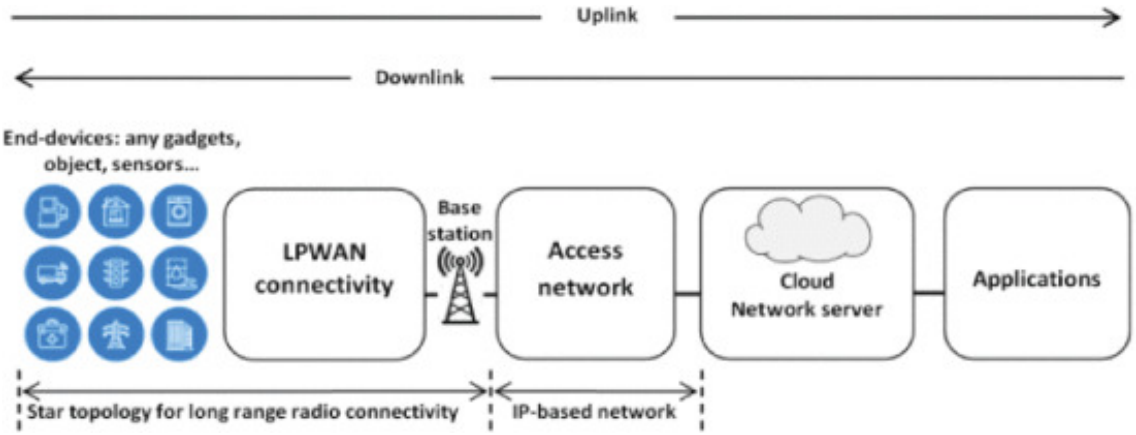
Wi-Fi-verkon yhteyspisteitä on yli kaksi miljardia, joten verkko on saatavilla lähes kaikkialla, jossa M2M-dataa kannattaa kerätä. Zigbeehen verrattuna Wi-Fi kuluttaa datansiirtoon huomattavasti vähemmän energiaa, sillä dataa siirretään yhden hypyn kautta ainoastaan tarvittaessa. Radio voidaan sulkea, kun viestejä ei lähetetä. ZigBeessä tämä ei ole mahdollista, sillä solmujen pitää kuunnella säännöllisesti naapureita, mikäli ne haluavat lähettää viestejä. Wi-Fi-verkkojen ongelmana on kuitenkin suhteellisen lyhyt kantama tukiaseman ja sen kanssa kommunikoivan laitteen välillä, joten kantaman puute rajoittaa datan keräämistä laajalle alueelle asennetuista antureista. [1]

Puhelinverkkoja käytettäessä päästään eroon Wi-Fin kantavuusongelmasta, sillä yhdellä tukiasemalla voidaan kattaa useiden kilometrien säteinen alue. Matkapuhelimet kykenevät lähettämään paljon dataa ollessaan kaukana tukiasemasta. Toisaalta tämä kommukointi kuluttaa paljon tehoa ja siten tyhjentää puhelimen akun suhteellisen nopeasti. Matkapuhelimen käyttäjälle akun tyhjentyminen ei ole ongelma, sillä akun voi ladata uudelleen päivittäin. M2M-kommunikoinnissa oleellista on kuitenkin, että laite toimii samalla energianlähteellä asentamisensa jälkeen mahdollisesti vuosia, ja toisaalta laitteen lähettämät datamäärät ovat huomattavasti pienempiä. Tämä on johtanut kysyntään verkkotekniikoille, joissa voidaan kommunikoida laajan kantaman yli pienellä tehonkulutuksella. [1]

Vastauksena tarpeeseen on kehitetty joukko verkkotekniikoita, joita kutsutaan yhteisnimellä Matalan tehon laaja-alaiset verkot (engl. *Low Power Wide Area Network, LPWAN*). LPWAN-teknologioita yhdistää mahdollisuus yhdistää tuhansia laitteita samaan yhteyspisteeseen laajalla alueella. Tekniikoissa päästään matalaan tehonkulutukseen uhraamalla datansiirtonopeutta, mutta tämä ei ole ongelma laitteiden ollessa yksinkertaisia antureita, joiden on tarkoitus lähettää pieniä datamääriä pitkillä lähetysväleillä. [27]

Pitkän kantamansa ansiosta LPWAN-verkot voivat käyttää verkkotopologiana

tähtiverkkoa, eli laitteet ovat kytkeytyneenä suoraan julkiseen tukiasemaan. Verrattuna mesh-verkkoon (ZigBee), jolla voidaan kattaa laaja alue lyhyemmän kantaman verkoilla, tähtiverkko on yksinkertaisempi ja vähäenergisempi, koska laitteen ei tarvitse kuunnella naapureitaan ennen lähettämistä. Tukiasema on myös jatkuvasti päällä, joten datan lähetys on aina mahdollista. [22]



Kuva 2. *LPWAN-verkkotopologia [22]*

LPWAN-verkossa viestit voivat kulkea kahteen suuntaan, kuten on esitetty Kuvassa 2. Uplink-viestinnässä päätelaite lähettää viestin kohti tukiasemaa. Kun tukiasema vastaaonottaa viestin, se lähettää sen eteennpäin verkkopalvelimelle IP-perusteisessa verkossa. Verkkopalvelin tarkastaa viestien autenttisuuden ja lähettää edelleen sovelluspalvelimelle. Downlink-viestinnässä datan suunta on päinvastainen. Siinä sovellus lähettää viestin verkkopalvelimelle, joka puolestaan lähettää sen sopivalle tukiasemalle ja edelleen päätelaitteelle. Downlink-mahdollisuudesta huolimatta LPWAN-verkot on ensisijaisesti suunniteltu Uplink-viestintään. [22]

LPWAN-teknologat voidaan jakaa kahtia lisenssoitavia taajuuskaistoja käyttäviin tekniikoihin ja lisenssointia tarvitsemattomiin tekniikoihin. Ensin mainittua edustaa Narrowband-IoT ja LTE Cat-M1, jälkimmäistä muun muassa LoRa ja Sigfox. Lisenssoidun kaistan etuna on, ettei kyseisellä taajuusalueella saa olla muuta verkkoliikennettä, mikä lisää verkon luotettavuutta vähentäen häiriöitä. Lisensoinnin haittapuolena on, että verkon ylläpitämisestä pitää maksaa lisensointimaksuja, kun taas lisensoimattoman verkon ylläpitämien on ilmaista. [27]

2.2.1 LoRa

LoRa on Semtech-nimisen yhdistyksen kehittämä ja valmistama LPWAN-teknologia. LoRa Allianceen kuuluu noin 400 jäsenyritystä ympäri maailmaa. LoRa:lla tarkoitetaan verkkomallin fyysistä kerrosta eli piiriä, joka mahdollistaa radioyhteyden laitteiden välillä. LoRa on suunnattu pääasiassa sovelluksiin, joissa data siirtyy Upstreamin mukaisesti anturilta yhteyspisteelle (engl. *gateway*). Toisaalta kommunikoivia laitteita voi olla hyvin useita. [10]

LoRa käyttää *Chirp Spread Spectrum* (CSS) -nimistä modulaatiotekniikkaa. CSS jakaa lähetettävät viestit usealle taajuusalueelle ja datan siirtonopeuksille vähentäen näin riskiä useiden viestien yhteentörmäyksistä. Vähentyneet yhteentörmäykset kasvattavat yhteyspisteen kapasiteettia. Lisäksi CSS tarjoaa suuremman vahvistuksen, mikä mahdollistaa pienemmän lähetystehon samalla radiolinkkibudjetilla. LoRa käyttää taajuuskaistaa 433 MHz Aasiassa, 868 MHz Euroopassa ja 915 MHz USA:ssa. [10]

Long Range Wide Area Network (LoRaWAN) on Lora Alliancen säätelemä langaton kommunikointistandardi, joka koostuu LoRa:sta, sen päällä toimivasta MAC-kerroksesta ja sovellusstandardeista. LoRaWAN tarjoaa seuraavanlaisia ominaisuuksia:

- Suuri kantama (5 km kaupunkialueella, yli 10 km esikaupunkialueella ja yli 80 km ideaalisissa olosuhteissa)
- pitkä akunkesto (10 vuotta)
- matala yksikköhinta (alle 5 €/moduuli)
- matala datansiirtonopeus (0.3 bps - 50 kbps, tyypillisesti 10 kB/päivä)
- salattu yhteys
- toimii lisensoimattomalla taajuuskaistalla
- kaksisuuntainen tiedonsiirto

Ominaisuuksista huomataan, että ainoastaan datansiirtonopeus näyttää huonolta, mutta muut ominaisuudet loistavat. Langattomassa tiedonsiirrossa joudutaan tekemään kompromissi tehonkulutuksen, etäisyyden ja datansiirtonopeuden välillä. LoRaWAN on suunniteltu käyttökohteisiin, jossa riittää matala datansiirtonopeus.

LoRaWAN tukee kolmea luokkaa (engl. *Class*), jotka vaikuttavat laitteen energiankulutukseen. A-luokka säästää eniten energiaa. Siinä päätelaite on aina unessa, kunnes lähetyksen laukaisee jokin sen antureista. Lora-palvelin voi ainoastaan vastata päätelaitteen lähetykseen, mutta muuten Downlink ei ole käytössä. Hyviä sovelluksia A-luokan laitteille ovat erilaiset hälyttimet, kuten palvovaroitin. B-luokkaa käytettäessä yhteyspiste kertoo päätelaitteelle ajanhetketket, jolloin se voi lähettää dataa (Uplink). Muutoin se nukkuu. Palvelin voi kommunikoida päätelaitteen kanssa jokaisen Uplink-viestinnän jälkeen. C-luokassa päätelaite kuuntelee jatkuvasti, mikä mahdollistaa viiveettömän kommunikoinnin molempiin suuntiin. Samasta syystä C-luokka kuluttaa eniten energiaa. [10]

2.2.2 Sigfox

Sigfox on vuonna 2009 perustettu ranskalainen yritys, joka on onnistunut markkinoimaan omaa LPWAN-verkkoaan Euroopassa. Sigfox on yksityinen verkkoratkaistu, jossa hitaalla modulaationopeudella saavutetaan suuri kantama. Sigfoxin modulaatio tarvitsee hyvin kapean taajuuskaistan. Tarvittava

kokonaiskaistaleveys on 200 KHz, jonka sisällä yksittäisen viestin lähettämiseen riittää 100 Hz:n kaistaleveys. Datansiirtonopeus vaihtelee 100–600 bps ja kantama 10–30 km olosuhteista riippuen. Verkko on saatavilla 32 maassa. [10]

Sigfox on erittäin kevyt protokolla, joten se käsittelee pienet, maksimissaan 12 tavun hyötykuorman viestit erittäin energiatehokkaasti. Akunkesto saadaan pitkäksi, koska protokollassa hyötykuorman lisäksi lähetetään vain vähän ylimääräistä. Sigfox rajoittaa päivässä lähetettävät Uplink-viestit 140:een ja Downlink-viestit 4 maksimissaan 8 tavun hyötykuormalliseen viestiin. Koska jokaista lähetettyä viestiä kohden ei voida vastata kuittauksella, päätelaite lähettää saman Uplink-viestin kolme kertaa eri taajuuskanavilla. [22]

Sigfox on LoRan tavoin hyvä vaihtoehto sovelluksille, joissa datan lähetys on satunnaista ja lähetettävät datamäärät pieniä. Haittapuolena ratkaisussa on erittäin rajoittunut mahdollisuus lähettää dataa takaisin antureille. Myös johtuen taajuuskaistan lisensoimattomuudesta singnaalin häirintä saattaa muodostua ongelmaksi. [10]

2.2.3 LTE Cat-M1

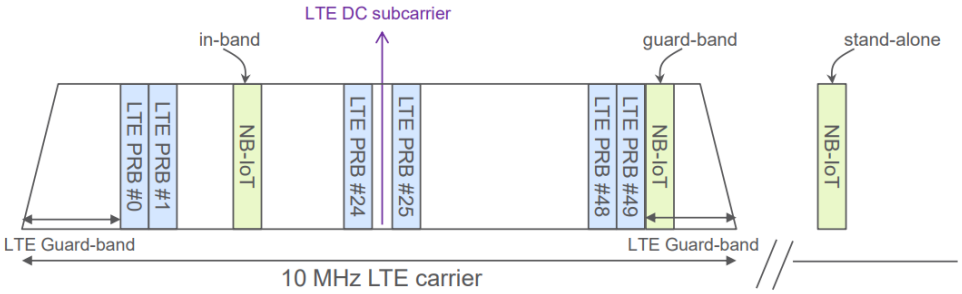
Mobiiliverkkoa standardoivan *Third Generation Partnership Project*in (3GPP) 13. julkaisussa on esitelty laajan IoT-verkon kannalta lukuisia avainominaisuuksia. Julkaisu keskittyy tehostamaan olemassa olevia *Global System for Mobile Communications* (GSM)- ja *Long-Term Evolution* (LTE) -verkkoja. LTE Cat-M1 -standardilla optimoidaan olemassa olevan LTE-verkon resurssienkäyttöä ja siten vähennetään energiankulutusta. [35]

LTE Cat-M1 -verkossa datansiirtonopeus molempiin suuntiin on 375 kb/s, joten tekniikka sopii sovelluskohteisiin, joissa voidaan siirtää myös muita LPWAN-tekniikoita suurempia datamääriä. Suuremmat datamäärät mahdollistavat päätelaitteen ohjelmiston päivittämisen langattomasti järkevissä aikaikkunoissa, joten teknologia soveltuu hyvin kriittisempiin sovelluksiin, joissa päätelaitteet asennetaan kohteeseen pitkiksi ajoiksi. Matalilla datansiirtomäärillä päästään kymmenen vuoden akunkestoon, mikä vähentää ylläpitokustannuksia. LTE Cat-M1 on ideaalinen liikkuviin käyttökohteisiin, sillä päätelaite vaihtaa siinä mobiilitukiasemaa samoin kuin nopeassa LTE-verkossa. Teknologia tukee myös *voice over LTE* (VoLTE) -toiminnallisuutta, josta voi olla hyötyä sovelluksissa, jotka vaativat vuorovaikutusta ihmisen kanssa. [27]

2.2.4 NB-IoT

3GPP:n 13. julkaisu esittelee LTE-verkon optimoinnin lisäksi uuden LPWAN-teknologian nimeltä *Narrowband Internet of Things* (NB-IoT). NB-IoT on uusi radiotekniikka siinä mielessä, ettei se ole yhteensopiva vanhojen 3GPP-laitteiden kanssa. Se on kuitenkin suunniteltu hyvään rinnakkaiseloon olemassa olevien GSM- ja

LTE-verkkojen kanssa. NB-IoT tarvitsee toimiakseen 180 kHz:n kaistaleveyden sekä lähetykseen (Uplink) että vastaanottoon (Downlink), joten GSM-operaattori voi korvata yhden kantoaallon NB-IoT:lla, tai LTE-operaattori voi luovuttaa yhden resurssilohkoistaan (engl. *Physical Resource Block*, PRB) sille. [35]



Kuva 3. *NB-IoT:n eri asennusvaihtoehdot [35]*

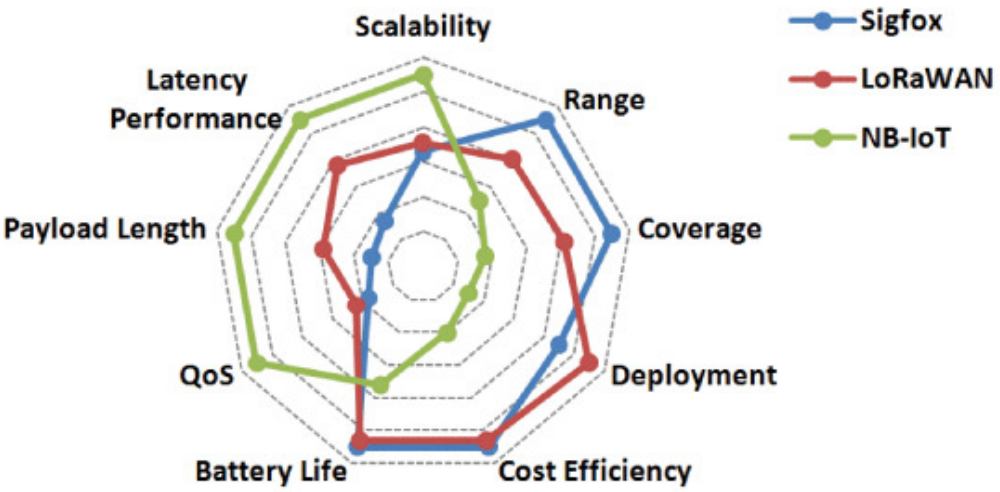
NB-IoT:n käyttöönotossa on kolme vaihtoehtoa. Se voidaan asentaa oman (stand-alone) kantoaallon päällä, kunhan sillä on käytössään yli 180 kHz:n kaistanleveys, tai sitten se voidaan asentaa LTE-spektrin sisään joko kaistalle (in-band) tai kaistojen väliin (guard band), kuten Kuvassa 3 on esitetty. Kaistan sisälle toimitettaessa LTE luovuttaa NB-IoT:lle jonkun resurssilohkoistaan. LTE-kantoaaltojen reunoissa on käyttämätöntä taajuusaluetta, joten NB-IoT voidaan asentaa myös sinne. Asennustavan pitäisi kuitenkin olla laitteelle läpinäkyvä, kun se käynnistetään ensi kertaa ja se etsii kantoaaltoa. [35]

NB-IoT uudelleenkäyttää suurta osaa LTE-tekniikasta, mikä vähentää spesifikaatioiden määritykseen ja tuotekehitykseen käytettävää aikaa. NB-IoT supistaa LTE:n ominaisuudet minimiin ja tehostaa IoT:n kannalta olennaisia ominaisuuksia. Esimerkiksi normaalissa LTE-verkossa tukiasema lähettää kaikille solun laitteille yhteistä tietoa, mutta NB-IoT:ssa kyseinen lähettäminen on rajattu minimiin virran säästämiseksi. Verkkotekniikka on optimoitu pienien ja harvoin lähetettävien viestien lähetykseen ja saatu energia- ja kustannustehokkaaksi luopumalla kaikista IoT:n kannalta tarpeettomista LTE:n ominaisuuksista. [22]

NB-IoT mahdollistaa 100 000 laitteen kytkemisen yhteen soluun, ja määrää voidaan vielä kasvattaa varaamalla sille useampi kantoaalto. Se käyttää QPSK-modulaatiota ja taajuuskanavointia (engl. *Frequency Division Multiple Access*, FDMA) Uplink-viestinnässä ja useamman laitteen versiota taajuuskanavoinnista (engl *Orthogonal FDMA*, OFDMA) Downlink-viestinnässä. Tämä mahdollistaa 20 kbps:n Uplink- ja 200 kbps:n Downlink-nopeuden 1600 tavun viestikohtaisella maksimihyötykuormalla. NB-IoT ja LTE-M voivat molemmat saavuttaa 10 vuoden akunkeston lähettämällä keskimäärin 200 tavua päivässä. [22]

NB-IoT:n hyötyjä ja haittoja muihin LPWAN-tekniikoihin on havainnollistettu Kuvassa 4. Lisensoidun taajuuskaistan ja LTE:hen pohjautuvan synkronisen protokollan vuoksi se tarjoaa ainoana tietoliikenteen priorisointia (engl. *Quality of service*, QoS). NB-IoT:n

viestinnässä lähetysviive on myös LoRaWAN:ia ja Sigfoxia pienempi. Kaikki tekniikat tarjoavat korkean skaalautuvuuden. NB-IoT mahdollistaa kuitenkin 100 000 laitteen kytkemisen tukiasemaan, kun LoRaWAN ja Sigfox tukee 50 000 laitetta tukiasemaa kohden. NB-IoT-datapaketin hyötykuorman maksimikoko on 1600 tavua, mikä on huomattavasti suurempi kuin LoRaWAN:n 243 tavua tai Sigfoxin 12 tavua. [22]



Kuva 4. *NB-IoT:n hyödyt ja haitat LoRaWAN:iin ja Sigfox:iin nähden [22]*

NB-IoT häviää erityisesti kustannustehokkuudessa, sillä päätelaitteiden hinnat maksavat yli 20 euroa, kun vastaava kustannus LoRaWANissa on alle viisi ja Sigfoxissa alle kaksi euroa. NB-IoT:ssa taajuuskaistasta joutuu myös maksamaan lisenssimaksuja niiden ollessa ilmaisia kahdella muulla tekniikalla. Myös tukiasemien asentaminen maksaa NB-IoT-verkolle eniten. NB-IoT:n synkronoitu viestintä ja hyvä QoS aiheuttavat ylimääräistä virrankulutusta ja taajuuskanavointi vaatii suurempia virtapiikkejä, joten NB-IoT:n akunkesto on kahta muuta tekniikkaa huonompi. NB-IoT häviää kahdelle muulle tekniikalle myös kantamassa ja uutena tekniikkana sitä asennetaan vasta koekäyttöön. LoRaWAN on mahdollista asentaa myös halpana paikallisverkkona tukiasemien sijaan. [22]

2.3 HTTP

Hypertext Transfer Protocol (HTTP) [15] on sovellustason protokolla, jota on käytetty laajasti tiedonsiirtoon internetissä. HTTP on RESTin mukainen pyyntö/vastaus (engl. *request/response*) -protokolla. Toisin sanoen asiakas lähettää palvelimelle kyselyitä, johon tämä vastaa. Perinteisessä IP-viestinnässä asiakas on HTTP-protokollaa käyttävä internet-selain. HTTP sisältää myös RESTin mukaisen resurssimallin, jossa dataobjektit identifioidaan *Uniform Resource Locator*in (URL) avulla, joka on URI:n alaluokka. URL:n yleinen muoto on seuraavanlainen [9]:

<Protokolla>://<Host>:<Portti>:<absoluuttinen polku>:?<Etsintäosa>

Tässä muodossa Protokolla (http) ja Host (palvelimen osoite) ovat vaadittuja. Absoluuttinen polku viittaa tiettyyn resurssiin, ja Etsintäosalla voidaan antaa parametrejä logiikalle, jota palvelin suorittaa kyselyn saapuessa.

HTTP-protokollassa on yleisimmin käytettynä Taulukossa 1 listatut kuusi metodia, joista yksi sisällytetään aina kyselyyn kertomaan palvelimelle, mitä resurssille tulisi tehdä. GET-metodilla asiakas pyytää resurssin sisältämää dataa palvelimelta. PUT- ja POST-pyynnöillä asiakas pyrkii kirjoittamaan serverille, ja serveri palauttaa onnistumisesta riippuvan statuskoodin. Vähemmän käytettyjä metodeja ovat HEAD, DELETE ja TRACE. Metodeilla on kaksi ominaisuutta, turvallinen ja idempotentti. Turvallisella tarkoitetaan sitä, ettei resurssin tila palvelimella muutu kyselyn yhteydessä. Idempotentti tarkoittaa, että saman kyselyn uudelleenlähettäminen tuottaa saman tuloksen. [24]

Taulukko 1. *Tavallisimmat HTTP-metodit*

Metodi	**Selitys**	**Turvallinen**	**Idempotentti**
GET	pyydä resurssia palvelimelta	kyllä	kyllä
POST	lähetä data palvelimelle prosessoitavaksi	ei	ei
PUT	tallenna resurssi palvelimelle	ei	kyllä
DELETE	poista resurssi palvelimelta	ei	kyllä
HEAD	pyydä pelkkää headeria ilman dataa	kyllä	kyllä
TRACE	seuraa pyyntöä palvelimelle	kyllä	kyllä

2.4 REST

REST (engl. *REpresantional State Transfer*) tarkoittaa sääntöjoukkoa, jonka noudattaminen johtaa hyväksi todettuun tietoliikennemalliin. REST on kehitetty rajoittamaan sovellusten kehitystä, jotta niiden ylläpito ja laajennettavuus pysyisivät yksinkertaisina. Ollakseen RESTin mukainen systeemin täytyy täyttää seuraavat ehdot:

- Client-Server-systeemi: kommunikaation pitää tapahtua kahden koneen välillä siten, että toinen koneista toimii palvelimena, jolle toinen lähettää kyselyitä.
- Tilaton: palvelimelle lähetettävät pyynnöt eivät saa riippua muista kyselyistä, vaan jokaisen kyselyn tulee olla itsenäinen.
- Sen pitää tukea välimuistiin tallentamista eri verkkokerroksissa.
- Saatavuus pitää olla yhdenmukainen: jokaisella resurssilla tulee olla uniikki osoite ja validi hakupolku.
- Kerroksittainen: sen pitää tukea laajennettavuutta
- Koodin tarjoaminen pyydettäessä. Tämä rajoite on vapaavalintainen, mutta mahdollistaa laajennettavuuden ajoaikana.

Nämä rajoitteet eivät kuitenkaan määrää käytettävää teknologiaa, vaan määrittävät, miten datan pitää siirtyä komponenttien välillä. Täten REST voidaan toteuttaa millä tahansa

verkkoarkkitehtuurilla. Suuri RESTiä noudattava systeemi on *World Wide Web* (WWW). [31]

RESTin mukaisessa arkkitehtuurissa kaikkea, jolla on osoite, kutsutaan resurssiksi. Resursseja voidaan siirtää palvelimen ja asiakaskoneen välillä. Esimerkkejä resursseista ovat lämpötila tiettynä kellonaikana tietyssä paikassa tai Googlen hakutulos tietylle hakusanalle. Eri kyselyt voivat palauttaa saman datan, vaikka resurssien osoitteet ovatkin itsenäisiä. Esimerkiksi kysely ”tiedoston viimeisin versio” ja ”tiedoston versio 12” palauttavat saman vastauksen mikäli 12. versio on uusin versio. [31]

Palvelimen ja asiakkaan välillä siirtyvää dataa kutsutaan resurssin kuvaukseksi. Kuvaus sisältää resurssin tilan muistissa kyselyhetkellä. Yleisesti kuvaus koostuu varsinaisesta datasta sekä metadatasta, joka kertoo, miten varsinainen data tulee prosessoida. Sama resurssi voi saada kyselystä riippuen erilaisia kuvauksia. Kuvaus voi olla esimerkiksi kuva, tekstitiedosto, XML-tiedosto, tai HTML-sivu, mutta se on silti saatavissa saman URI:n kautta. [31]

Uniform Resource Identifier (URI) on hyperlinkki resurssiin. URI mahdollistaa kuvausten vaihdon palvelimen ja asiakkaan välillä. RESTin mukaisessa systeemissä URIn ei ole tarkoitus muuttua ajan kuluessa, mikäli esimerkiksi palvelimen muistilaite vaihtuu. Ilman RESTin rajoitteita resursseihin päästään käsiksi tallennuspaikan avulla. Tällöin tiedoston uudelleennimeäminen muuttaa myös URIa. [31]

REST sisältää neljä toimenpidettä, joilla resursseja voidaan käsitellä. Metodit ovat Create, Retrieve, Update ja Delete (CRUD). CRUD-toiminnnot voidaan yhdistää HTTP-metodeihin Taulukon 2 mukaisesti. Kun asiakasohjelma haluaa saada resurssin kuvauksen palvelimelta, se lähettää sille GET-pyynnön. Palvelin lähettää vastauksen, jonka hyötykuormana on resurssin kuvaus, tai epäonnistumisen, mikäli pyynnön käsittelyssä tapahtuu virhe. Mikäli asiakasohjelma haluaa luoda palvelimelle uuden resurssin, se lähettää sille POST-pyynnön. Resurssin päivitykseen asiakas käyttää vastaavasti PUT-pyyntöä ja poistamiseen DELETE-pyyntöä. [31]

Taulukko 2. *CRUD HTTP-metodeina.*

Toimenpide	**HTTP-metodi**
CREATE	POST
RETRIEVE	GET
UPDATE	PUT
DELETE	DELETE

2.5 CoAP

Constrained Application Protocol (CoAP) on erityisesti rajoitetuille verkoille ja verkkolaitteille suunnattu datansiirtoprotokolla. Rajoitetuissa verkoissa datan siirtomäärät verkon läpi ovat pieniä, ja datapakettien siirroissa tapahtuu paljon virheitä.

Rajoitetulla verkkolaitteella on yleensä 8-bittinen mikroprosessori pienellä muistilla ja rajallisella energianlähteellä. CoAP on suunniteltu M2M-sovellusta, kuten rakennusautomaatiota, varten. [11]

Coap tarjoaa REST:n mukaisen request/response-mallin päätelaitteiden välille, tukee sisäänrakennettua resurssien löytämistä ja sisältää webin keskeisiä konsepteja, kuten URI:n ja internetin mediatyypit. Samankaltaisuuksien vuoksi CoAP:n ja HTTP:n välille on helppo rakentaa rajapinta, jolla saadaan rajoitettu laite integroitua verkkoon samalla huomioiden erityisvaatimukset, kuten pieni viestin koko ja yksinkertaisuus. CoAP sisältää seuraavat ominaisuudet:

- M2M-vaatimukset rajoitetuissa ympäristöissä täyttävä verkkoprotokolla
- UDP:n käyttö kuljetuskerroksessa, viestinvaihdon luotettavuus on vapaavalintaista
- Asynkroninen viestien vastaanotto
- Helposti parsittava kompakti viestiformaatti
- URI-tuki
- Yksinkertaiset välityspalvelin- ja välimuistiominaisuudet
- Tilaton yhteensopivuus HTTP-viestien kanssa, mikä mahdollistaa yksinkertaisen muunnoksen protokollasta toiseen välityspalvelimessa.
- Salaus Datagram Transport Layer Security (DTLS) -protokollalla

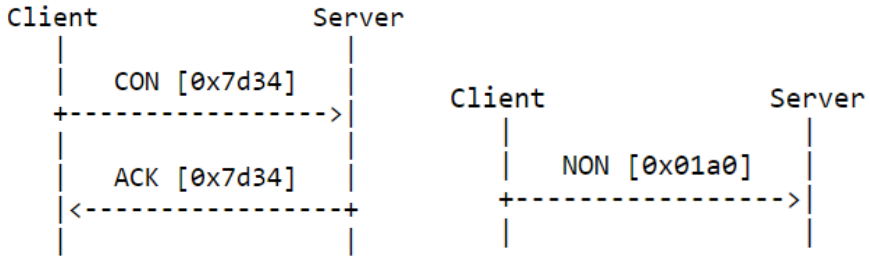
2.5.1 Viestienvaihtomalli

CoAP:n viestienvaihto tapahtuu UDP-yhteyden yli. Viestit ovat lyhyitä sisältäen lyhyen otsikkokentän, valintoja ja varsinaisen viestin. Tämä formaatti on käytössä sekä kyselyillä että vastausviesteillä. Viestit sisältävät lyhyen tunnisteen (ID), jolla tunnistetaan kopiot samasta viestistä ja jota voidaan käyttää haluttaessa luotettavuuden lisäämiseen eli viestien perille pääsyn varmistamiseen. Kompakti viestikoko mahdollistaa 250 viestin lähetyksen sekunnissa oletusparametreilla. [11]

Luotettava viestienvaihto toteutetaan Kuvan 5 (a) mukaisesti. Asiakaspäätelaite merkitsee viestin vahvistettavaksi (*engl. Confirmable* CON), ja palvelin vastaa viestiin kuittauksella (*engl. Acknowledgement* ACK). Asiakas toistaa viestin lähetystä eksponentiaalisesti kasvavan viiveen välein, kunnes saa palvelimelta viestitunnisteen (Kuvan tapauksessa 0x7d34) sisältämän kuittauksen. Mikäli palvelin ei kykene prosessoimaan viestiä, se vastaa kuittauksen sijaan Reset-viestillä (RST). [11]

Luotettavan viestinvaihdon toteutustavasta voidaan havaita, että sama viesti saatetaan joutua lähettämään monta kertaa, mikäli viesti ei saavu perille, kuittaus ei saavu perille tai viestien vaihdossa kestää liian kauan. Jokainen viestin uudelleenlähetys luonnollisesti kuluttaa sähköä, ja siksi tapauksissa, joissa viestin perillepääsy ei ole kriittistä, kannattaa

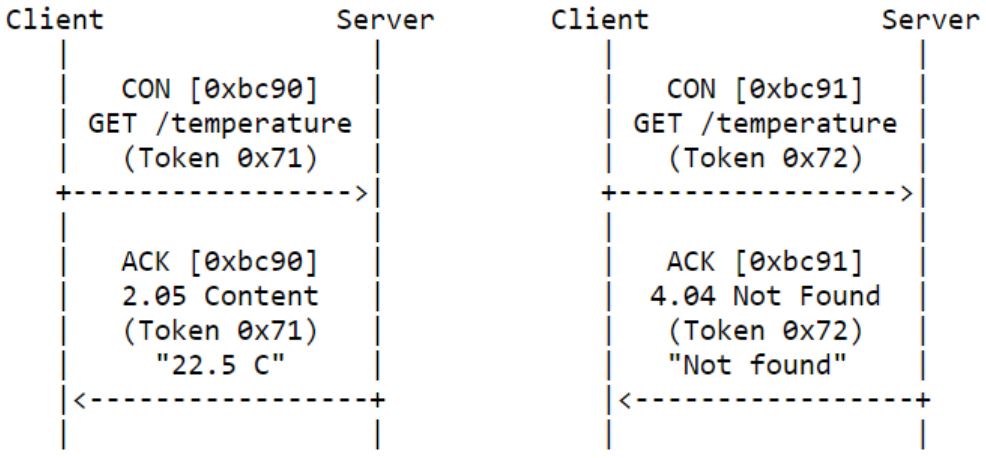
käyttää epäluotettavaa viestimistä, joka on esitetty Kuvassa 5 (b). Tällainen tilanne voi olla esimerkiksi yksittäisen mittausarvon lähetys. Asiakaspäätelaite merkitseen viestin vahvistamattomaksi (*engl. Non-Confirmable* NON) ja palvelin ei vastaa viestiin kuittauksella, mutta saattaa kuitenkin vastata RST-viestillä. Myös epäluotettavaan viestimiseen tarvitaan tunnisteet kopioiden havaitsemiseksi. [11]



(a) *Luotettava* ***(b)*** *Epäluotettava*

Kuva 5. *CoAP-protokollan viestienvaihtomalli [11]*

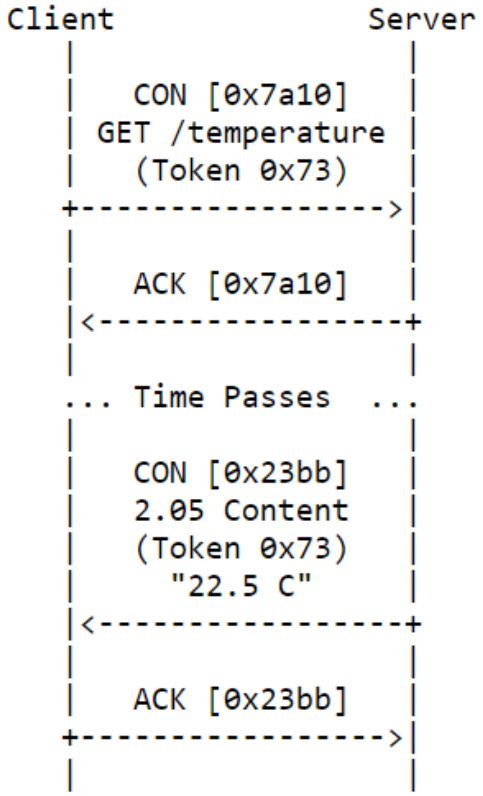
CoAP-viestissä esiintyy metodikoodi (GET, PUT, POST, DELETE) tai vastauskoodi riippuen viestistä. Viestin optioihin sisällytetään viestin tiedot, kuten URI, ja viestin mediatyyppi. Lisäksi viestissä on Token-kenttä, jolla vastaus voidaan yhdistää oikeaan kyselyyn. Tokenia ei pidä kuitenkaan sekoittaa edellä mainittuun tunnisteeseen, jolla varmistetaan viestin luotettavuus. CoAP-metodikoodin sisältämä kysely voidaan lähettää joko CON- tai NON-viestinä. Mikäli palvelimella on vastaus valmiina, voidaan se sisällyttää ACK-viestiin niin sanottuna "reppuselkävastauksena"(*engl. piggypacked response*). Kuvassa 6 esitetään kaksi tilannetta, jossa vastaus on välittömästi saatavilla. [11]



Kuva 6. *Kaksi esimerkkiä "reppuselkävastauksista"GET-kyselyyn [11]*

Edellä kuvattu "reppuselkävastaus"ACK-viestissä edellyttää vastauksen välitöntä saatavuutta, mutta näin ei välttämättä aina ole. Tällainen tilanne on esitetty Kuvassa 7. Palvelin vastaa kyselyyn tyhjällä kuittauksella, jotta asiakas tietää viestin tulleen perille

ja voi siten lopettaa uudelleenlähetyksen. Myöhemmin, vastauksen valmistuttua, palvelin lähettää sen uutena CON-viestinä, jonka puolestaan asiakas kuittaa. Jos kysely on lähetetty alunperin NON-viestinä CON-viestin sijaan, ACK-viestiä ei lähetetä vaan palvelin lähettää vastauksen toisena NON-viestinä. [11]



Kuva 7. *Kysely erillisellä vastauksella [11]*

2.5.2 Viestiformaatti

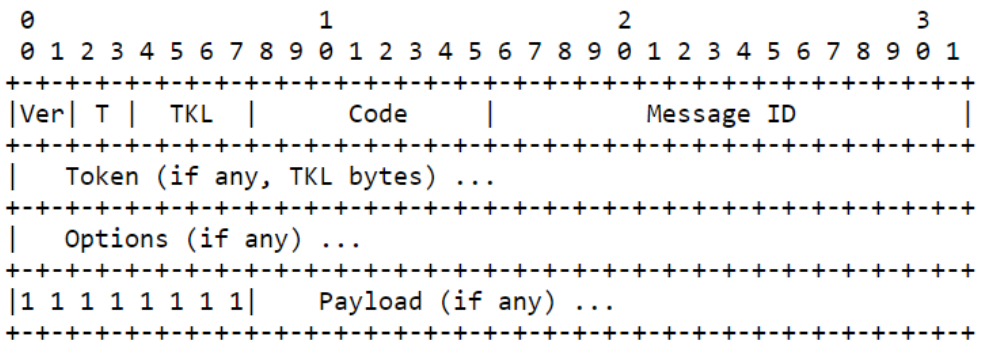
CoAP-viesteissä informaatio on tiivistetty siten, että jokainen yksittäinen viesti mahtuu yhteen UDP-pakettiin. UDP onkin CoAP:ssa käytetty oletusviestikerros. CoAP voidaan salata DTLS-salauksella, joka on UDP:n vastine TCP-yhteyksissä käytettävään TLS-salaukseen. Oletuksesta huolimatta CoAP-viestit voidaan tarvittaessa kuljettaa myös SMS:n tai TCP:n päällä. [11]

CoAP-viesti on koodattu Kuvan 8 mukaiseen binääriformaattiin. Viesti alkaa 4 tavun vakiopituisella otsikokentällä (*header*). Otsikkokenttä sisältää seuraavat osat:

- CoAP-versio (*Ver*)
- tyypin (*T*), eli onko kyseessä CON-, NON-, ACK- vai Reset-viesti
- otsikkoa seuraavan Token-kentän pituuden (*engl. Token length*, TKL)
- *Code*-kentän, joka indikoi, onko kyseessä kysely vai vastaus, ja sisältää lisäksi ensimmäisessä tapauksessa metodikoodin ja jälkimmäisessä vastauskoodin

- tunnisteen (*Message ID*), jota käytetään kopioiden havainnointiin sekä ACK- ja CON-viestien yhteenliittämiseen.

Otsikkoa seuraa pituudeltaan nollasta kahdeksaan tavuun vaihteleva Token-kenttä. Token-kentän arvolla CoAP-vastaus voidaan yhdistää oikeaan kyselyyn. Token-kentän perässä on sarja valintoja (engl. *Options*) tyyppi-pituus-arvo (engl. *type-lenght-value*, TLV) -formaatissa. Viestin lopussa on mahdollinen hyötykuorma (engl. *Payload*), joka täyttää lopun UDP-datagrammista. [11]



Kuva 8. *Viestiformaatti [11]*

2.5.3 Viestien lähetys

CoAP-viestien vaihto tapahtuu asynkronisesti päätepisteiden (engl. *endpoint*) välillä. Viestit saattavat saapua perille eri järjestyksessä kuin ne lähetettiin tai kadota matkalla. Samasta viestistä saattaa myös saapua perille useampi kopio. Tästä johtuen CoAP sisältää kevyen luottavuusmekanismin, kopiontunnistuksen ja viestien uudelleenlähetyksen eksponentiaalisesti kasvavan lähetysviiveen muodossa. [11]

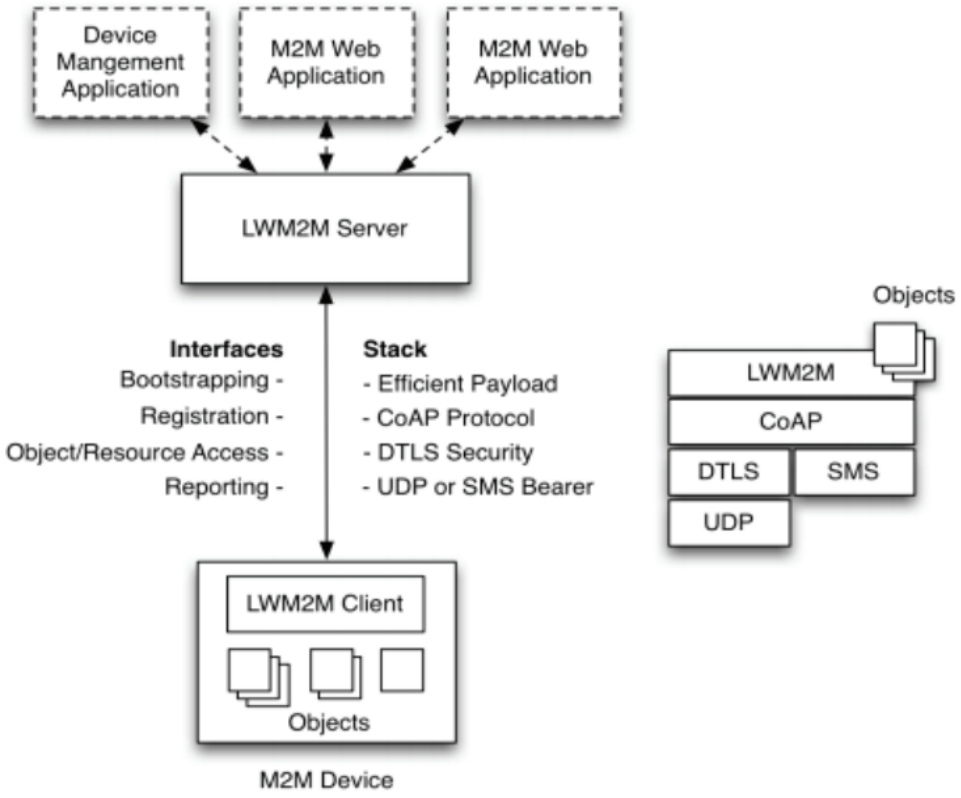
CoAP-päätepisteellä tarkoitetaan viestin lähettäjää tai vastaanottajaa. Päätepisteen identifiointi riippuu käytettävästä suojausmuodosta. Ilman suojausta identifiointi tapahtuu UDP-portin ja IP-osoitteen perusteella. CoAP määrittelee viestin koolle ylärajan siten, että viestin tulisi mahtua yhteen IP-pakettiin. Tällä vältetään yhden viestin jakaminen useaan lähetykseen. Mikäli viestin otsikkokentistä ei ole mitään tietoa, viestin koon ylärajaksi voidaan olettaa 1152 tavua, josta hyötykuormaa on 1024 tavua. [11]

Viestien ruuhkautumisen välttämiseksi asiakaspäätepisteen tulisi välttää useaa samanaikaikaista kyselyä tiettyyn palvelimeen. Asiakas lähettää samaan kyselyä eksponentiaalisesti kasvavan viiveen välein, kunnes saa palvelimelta vastauksen tai aika venyy niin pitkäksi, että asiakas luovuttaa. Kyselyn voidaan katsoa olevan voimassa koko tämän ajan, ja asiakkaan tulisi välttää uusien viestien lähettämistä ennen sen voimassaolon loppumista. [11]

2.6 LwM2M

Lightweight Machine to Machine (LwM2M) on *Open Mobile alliancen* (OMA) julkaisema järjestelmästandardi, jonka tarkoituksena on nopeuttaa asiakas-palvelinmallin omaavien rajoitetussa verkossa toimivien sovellusten tuotekehitystä. Standardi tarjoaa tehokkaan laitteenhallinatamallin sekä johdonmukaisen tavan tietoturvan toteuttamiseksi. LwM2M-spesifikaatio tarjoaa rajapinnat laitteen konfigurointiin, yhteyden seurantaan, tietoturvaan ja laitehojelmiston (engl. *Firmware*) päivittämiseen. LwM2M käyttää CoAP:ia viestien välittämiseen, mikä tekee siitä houkuttelevan protokollan IoT-sovelluksille. [28]

OMA LwM2M -arkkitehtuuri koostuu kahdesta osasta, kuten Kuvassa 9 on esitetty. LwM2M-palvelimen ja verkkosovellusten välillä on REST:n mukainen HTTP-rajapinta, ja lisäksi LwM2M-palvelimen ja -laitteen välinen kommunikaatio tapahtuu CoAP-protokollan yli. Ratkaisun kohderyhmä on pienen energiakulutuksen ja tehokkaan kaistankäyttötarpeen omaavat laitteet. Rajoitettujen laitteiden lisäksi LwM2M soveltuu luonnollisesti myös tehokkaammille laitteille, jotka hyötyvät tehokkaasta kommunikaatiosta. [28]



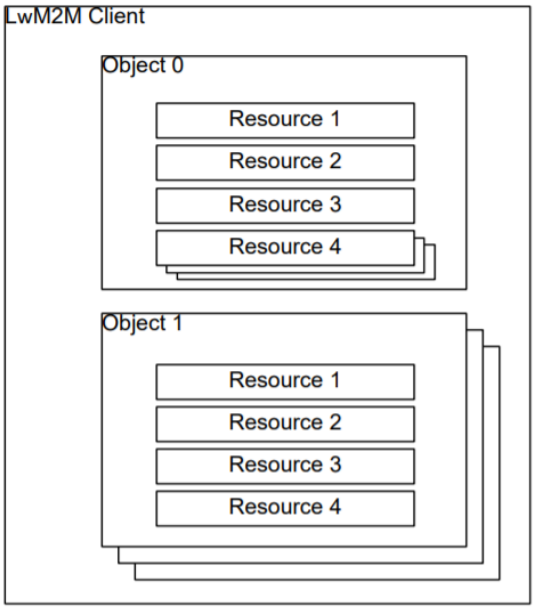
Kuva 9. *LwM2M-arkkitehtuuri [18]*

LwM2M-ratkaisu määrittää sovelluskerroksen kommunikaatioprotokollan asikasohjelman ja palvelimen välille. Palvelin sijaitsee tyypillisesti yksityisessä tai julkisessa datakeskuksessa, jossa sitä voi isännöidä verkkopalveluntarjoaja tai

sovelluspalveluntarjoaja. LwM2M-asiakasohjelma sijaitsee rajoitetussa laitteessa osana mikrokontrollerin ohjelmaa kirjastona tai moduulina. [28]

2.6.1 Resurssimalli

LwM2M-protokolla määrittää yksinkertaisen resurssimallin, jossa kaikki LwM2M-asiakasohjelman sisältämät tiedonpalaset on nimetty resursseiksi. Tällainen resurssi voi olla esimerkiksi lämpötila-anturin mittaama lämpötila numerona, korkein mitattu lämpötila tai lämpötilamittauksen yksikkö. Resurssit on järjestetty loogisiin kokonaisuuksiin, joita kutsutaan objekteiksi. Objekti voi olla esimerkiksi lämpötilaobjekti, jolloin se sisältää edellä mainitut lämpötilaan liittyvät resurssit. [18]



Kuva 10. *LWM2M-resurssimalli [28]*

Kuvassa 10 on esitetty LwM2M-protokollan resurssimalli. Kuvan esimerkissä asiakasohjelma sisältää 2 objektia, joissa kummassakin on 4 eri resurssia. Samasta objektista tai resurssista voi olla useampi instanssi. Kuvan esimerkissä objektista 1 on 3 instanssia ja objektin 0 resurssista 4 on myös 3 intanssia. Laitteessa voi olla esimerkiksi useampi lämpötila-anturi, jolloin jokaista anturia kohden on luotu oma instanssi lämpötila-objektista. [18]

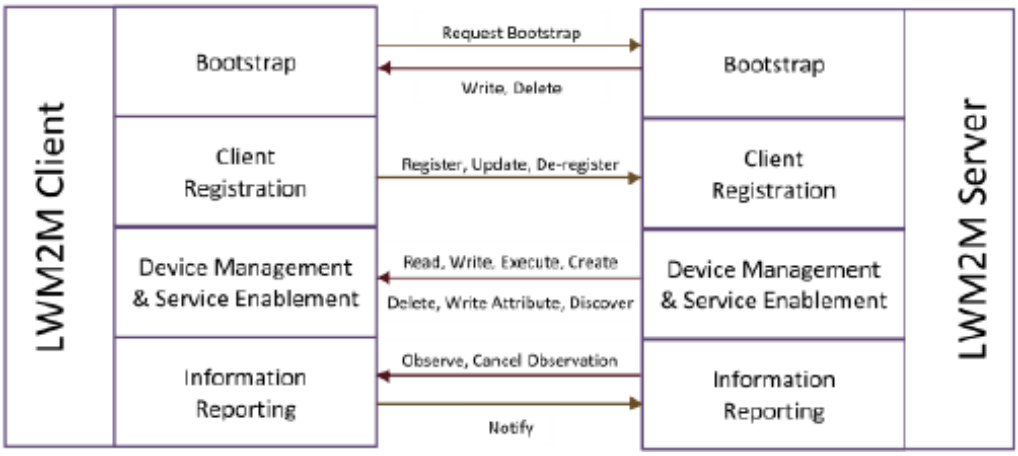
OMA Naming Authority (OMNA) määrittää standardiobjekteja ja -resursseja. Näiden lisäksi muut organisaatiot voivat määrittää omia räätälöityjä objekteja. Objektiin sisältyvät resurssit voivat sisältää arvon, jolloin resurssiin kohdistettava operaatio voi olla luku ja/tai kirjoitus. Vaihtoehtoisesti resurssi voi olla osoite, jonka avulla LmM2M-palvelin voi käskeä asiakasohjelmaa suorittamaan jonkin toiminnon. Lisäksi spesifikaatiossa on mahdolisuus määrittää, onko tietty resurssi pakollinen osa objektia, jolloin se pitää olla määritettynä kaikissa objektin instansseissa, tai vapaaehtoinen, jolloin se saattaa olla määritettynä jossakin instanssissa tai ei missään. [18]

Jokaisella resurssilla on yksiselitteinen tunniste objektin sisällä. Tunniste on numerokoodaus tietylle objektille tai resurssille. OMNA määrittää esimerkiksi laiteobjektin tunnisteeksi numeron 3. Objektin sisällä on esimerkiksi resurssi "sarjanumero", jonka tunniste on numero 2. Mikäli LwM2M-palvelin haluaa tietoa kyseisestä asiakasohjelman resurssista, voi tämä viitata siihen URI:lla 3/0/2. URI:ssa 0 on laiteobjektin instanssi, koska tyypillisesti asiakasohjelma sisältää vain yhden instanssin laite-objektista. [18]

2.6.2 Rajapinnat

Protokolla määritteleen asiakasojelman ja palvelimen välille 4 loogista rajapintaa, jotka on esitetty Kuvassa 11.

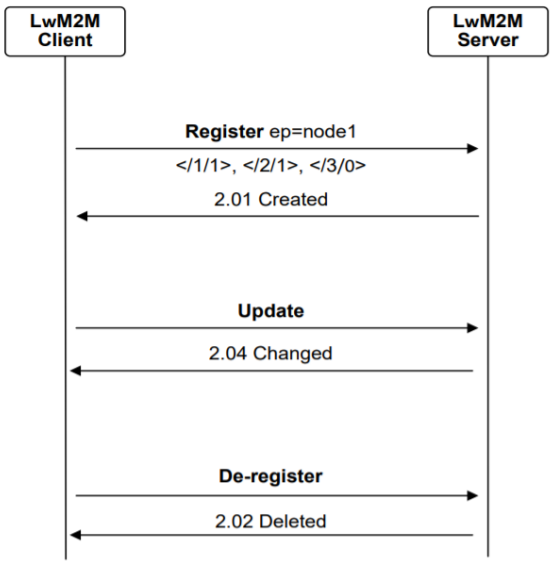
- Esilatausrajapinta (engl. *Bootstrap*) sisältää tietoturva-avainten- ja yhteydenhallintaa sekä laitteen kofigurointia varsinaselle palvelimelle kirjautumista varten.
- Laitteen paljastus- ja rekisteröintirajapinta (engl. *Device Discovery & Client Registration*) mahdollistaa sen, että palvelin saa tiedon laitteen olemassaolosta ja voi kirjata ylös laitteessa sijaitsevat resurssit.
- Laitteenhallinta ja palveluiden aktivointi (engl. *Device Management & Service Enablement*) tapahtuu LwM2M-palvelimen lähettämien pyyntöjen ja asiakaslaitteen vastausten seurauksena.
- Datan raportointirajapinta (engl. *Infromation Reporting*) mahdollistaa tiedon välityksen laitteelta palvelimelle joko tapahtumien seurauksena tai säännöllisesti tietyn ajanjakson välein.



Kuva 11. *LwM2M-rajapinnat [28]*

Esilatausrajapintaa käytetään varustamaan LwM2M-asiakasohjelmatiedoilla, jota se tarvitsee myöhemmin LwM2M-palvelimelle rekisteröitymiseen. Rajapintaa käytettäessä LwM2M-asiakasohjelma aloittaa kommunikoinnin ”Bootstrap”-pyynnöllä, jonka jälkeen LwM2M-esilatauspalvelin kirjoittaa laitteeseen tarvittavat tiedot. Lopuksi palvelin

ilmoittaa laitteelle operaatioiden valmistumisesta ”Bootstrap finish” -viestillä. Esilataus voidaan suorittaa toisaalta myös laitteen omasta muistista tai muistikortilta. [18]

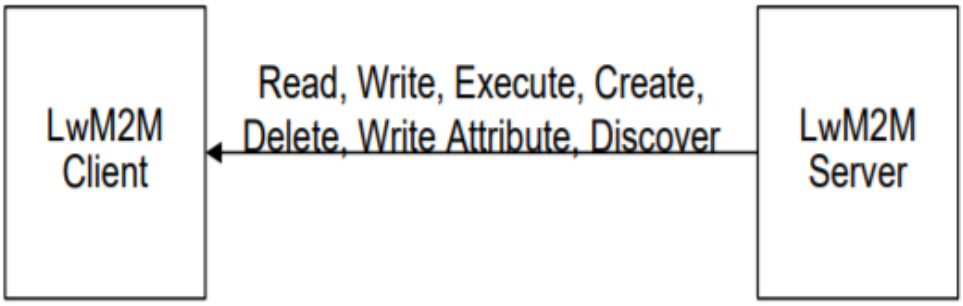


Kuva 12. *LwM2M-esimerkki sekvenssidiagrammi kirjautumisesta, päivityksestä ja uloskirjautumisesta [18]*

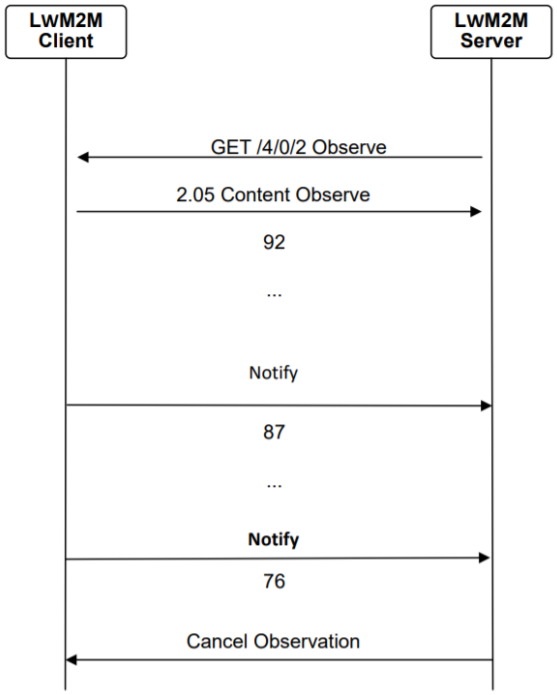
Rekisteröintirajapintaa käytetään laitteen palvelimelle rekisteröintiin, siis ilmoittamaan, että laite on olemassa, ja sille voidaan lähettää kyselyitä. Rajapinta sisältää 3 toimintoa: kirjautuminen, kirjautumisen uusiminen ja uloskirjautuminen, jotka on esitetty Kuvassa 12. Kirjautuminen tapahtuu LwM2M-asiakasohjelman suorittaessa ”register”-operaation, jossa se tarjoaa palvelimelle tarpeelliset tiedot asiakkaan kanssa kommunikointiin, istunnon ja rekisteröinnin ylläpitämiseen, sekä tiedot asiakaan sisältämistä resursseista. Istunnon ylläpitämiseksi laite ilmoittaa aikaikkunan, jonka sisällä asiakasohjelman tulee suorittaa kirjautumisen uusiminen, jolla laite ilmoittaa edelleen olemassaolostaan. Tätä operaatiota suoritetaan jaksollisesti, niin kauan kuin asiakasohjelma haluaa pysyä kirjautuneena. Mikäli asiakasohjelma ei lähetä uudelleenkirjautumista annetussa ajassa, palvelin kirjaa asiakkaan automaattisesti ulos ja "register-operaatio pitää suorittaa uudelleen, mikäli asiakasohjelma haluaa uusia istunnon. Asiakasohjelma voi myös itse kirjata itsensä ulos uloskirjautumisviestillä. [18]

Laitteenhallinta- ja palveluidentarjontarajapinta, joka on esitetty Kuvassa 13, sisältää operaatiot lue (engl. *Read*), kirjoita (engl. *Write*), suorita (engl. *Execute*), tuhoa (engl. *Delete*), luo (engl. *Create*), kirjoita lisämääreitä (engl. *Write-Attributes*) ja paljasta (engl. *Discover*). Rajapinnan kautta LwM2M-palvelin voi vuorovaikuttaa asiakasohjelmassa sijaitsevien objektien ja resurssien sekä näiden lisämääreiden kanssa. Lue-operaatiolla voidaan lukea kyseisen ajanhetken arvoja laitteesta. Paljasta-operaatiolla saadaan selville, mitkä resurssit tietystä objektista on toteutettu sekä mitä lisämääreitä näihin sisältyy. Lisämääreet ilmoittavat, montaako instanssia kyseinen objekti tai resurssi tukee tai millaisia asetuksia datan raportoinnille on asetettu. Kirjoita-operaatiolla laitteen arvoja voidaan päivittää. Lisämääreiden kirjoitusoperaatiolla voidaan muuttaa resurssien

datan raportointiin liittyviä lisämääreitä. Suorita-operaatiolla voidaan käskeä laitetta aloittamaan jokin toiminto. Luo- ja tuhoa-operaatiolla voidaan luoda uusia instansseja sekä poistaa niitä. [18]



Kuva 13. *Laitteenhallinta- ja palveluidentarjontarajapinta [18]*



Kuva 14. *Sekvenssikaavio datan keräämisestä [18]*

LwM2M-palvelin käyttää datan raportointirajapintaa nimensä mukaisesti datan keräämiseen asiakasohjelmalta. Kuvan 14 sekvenssikaaviossa esitetään, miten rajapintaa käytetään. Informaation kerääminen tietystä resurssista aloitetaan palvelimen tarkkailu-pyynnöllä (engl. *observe*). Tämän seurauksena asiakasohjelma alkaa lähettää dataa ilmoitus-viesteinä (engl. *notify*) aina resurssin arvon muuttuessa tai tarkkailuviestin sisältämien vapaavalintaisten, kontrolloivien lisämääreiden ehtojen täyttyessä. Lisämääreissä voidaan määrittää esimerkiksi minimiaika, jonka pitää kulua yksittäisten ilmoitusviestien välillä. Tällöin lämpötilamittauksessa tapahtuva muutos ei automaattisesti aiheuta uuden viestin lähetystä. Toisaalta lisämääreillä voidaan määrittää maksimiaika, jonka sisällä asiakasohjelman on raportoitava, vaikkei datassa

tapahtuisikaan muutoksia. Myös muutoksen suuruudelle voidaan asettaa kynnys, jonka ylittyessä asiakasohjelma lähettää ilmoituksen. Palvelin voi keskeyttää informaation keräämisen lähettämällä tarkkailun peruutus -viestin (engl. *Cancel Observation*) asiakkaalle, minkä jälkeen tämä ei enää lähetä uusia ilmoituksia kyseisestä resurssista. [18]

2.6.3 Salaus

LwM2M-protokolla perustuu CoAP-viestien siirtoon, joka puolestaan käyttää viestien siirtokerroksessa UDP- tai SMS-yhteyttä. LwM2M hyödyntää tämän vuoksi DTLS-protokollaa päätepisteiden autentikoimiseen ja datan salaukseen. Jotta salattu yhteys voidaan muodostaa, pitää asiakasohjelmaan sisällyttää tunnistautumistietoja. Nämä tiedot voidaan sisällyttää laitteen ohjelmaan valmistusvaiheessa tai laite voi hankkia ne esikommunikointirajapinnan kautta. LwM2M tukee kolmea eri tunnistautumisvaihtoehtoa:

- sertifikaatteja
- raakoja julkisia avaimia
- etukäteen jaettuja salaisuuksia

DTLS on UDP-yhteyksissä käytettävä viestien salausratkaisu, joka vastaa TCP-yhteyksissä käytettävää TLS-salausta. DTLS poikkeaa TLS-salauksesta ottamalla huomioon UDP-protokollalle tyypilliset datapakettien väärän saapumisjärjestyksen ja mahdollisen katoamisen ennen perille pääsyä. DTLS tarjoaa turvallisen kättelyviestien vaihtomekanismin ja istuntokohtaisten salausavainten luomisen, osapuolten autentikoinnin ja datan luotettavuuden varmistamisen. [18]

Salauksen toteuttamiseksi LwM2M-asiakasohjelmaan pitää sisällyttää tietoturvaobjekti. Objekti sisältää resurssit ”Security Mode”, joka määrittää mitä tietoturvatapaa käytetään, ”Public key or Identity”, joka edellisestä riippuen sisältää joko asiakasohjelman julkisen avaimen tai identiteetin, ”Server public key or identity”, joka sisältää palvelimen vastaavat tiedot ja ”Secred Key”, joka sisältää asiakasohjelman salaisen avaimen. [18]

2.6.4 Leshan

Leshan on Eclipsen avoimen lähdekoodin LwM2M-projekti. Projekti sisältää Java-toteutuksen LwM2M-palvelimesta sekä -asiakasohjelmasta. Leshan ei ole itsenäinen (engl. *standalone*) palvelintoteutus, vaan joukko kirjastoja, joita apuna käyttäen kukin voi rakentaa oman LwM2M-sovelluksen. Leshan tarjoaa kuitenkin esimerkkitoteutuksen palvelimesta ja asiakkaasta, joita voi käyttää testaustarkoituksiin ja pohjana oman sovelluksen luomisessa. [14]

Leshan-projekti on aloitettu vuonna 2014, ja se on rakennettu ainkaisemman Eclipse Wakaama CoAP -toteutuksen päälle. Leshan käyttää DTLS-salausta, joka on toteutettu

käyttäen Eclipsen Scandium-projektia. Projektin esimerkkipalvelin on käynnissä hiekkalaatikossa ”https://leshan.eclipseprojects.io”, johon voi kuka tahansa yhdistää laitteitaan ilmaiseksi osoitteessa ”coap://leshan.eclipseprojects.io:5683” nopean tuotekehityksen edistämiseksi. [14]

2.7 Bluetooth Low Energy

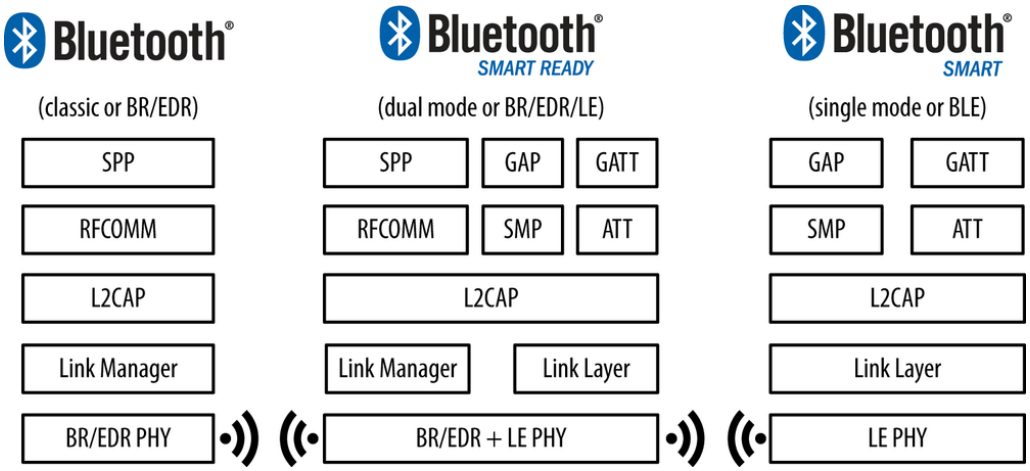
Bluetooth on globaali lyhyen kantaman langaton kommunikaatiostandardi, jossa laitteet viestivät radiolinkin yli. Se oli alunperin tarkoitettu korvaamaan sarjayhteyttä käyttävät datakaapelit. Ajan myötä käyttökohteet ovat laajentuneet tietokoneiden välisestä tiedostojen vaihdosta musiikin kuunteluun ja puheluiden vastaamiseen Bluetooth-kuulokkeilla. Bluetooth löytyy tänä päivänä käytännössä kaikista markkinoilla olevista matkapuhelimista, tableteista ja kannettavista tietokoneista. [16]

Bluetooth Low Energy (BLE tai *Bluetooth Smart*) on yksi Bluetooth-teknologian viimeisimmistä lisäyksistä, joka lisättiin standardin versioon 4.0. Teknologia on suunnattu erittäin pienen tehonkulutuksen omaaville laitteille, joten siinä energiantarvetta on pyritty vähentämään entisestään klassiseen Bluetoothiin verrattuna. Laitteiden odotetaan kykenevän useiden kuukausien tai jopa vuosien käyttöikään yhdellä nappiparistolla ilman pariston vaihtotarvetta. Datan siirto tapahtuu harvaan lähetettävinä purskeina, joten BLE sopii sovelluksiin, jotka eivät vaadi suuria siirtonopeuksia tai datan suoratoistoa. [16]

BLE-lisäyksen jälkeen Bluetooth-standardin voidaan katsoa sisältävän 2 eri tekniikkaa, jotka jakavat osia protokollakerroksista. Tekniikat eroavat kuitenkin toisistaan riittävästi, etteivät ne ole keskenään yhteensopivia. Klassista Bluetoothia tukevaan laitteeseen viitataan lyhenteellä BR/EDR. Bluetooth-laite voi tukea vain toista spesifikaation tekniikoista, jolloin sen konfiguraatio on single-mode. Laite voi tukea myös molempia tekniikoita, jolloin sen konfiguraatio on dual-mode ja laitteesta voidaan käyttää termiä Smart Ready. Kuva 15 esittää eri konfiguraatioiden mahdollisuuksia ja niiden sisältämien protokollakerrosten eroja. [13]

Bluetooth-laitteen voidaan katsoa koostuvan kolmesta osasta, jotka ovat käyttökohdekohtainen sovellus, Host ja Controller. Host kattaa protokollan ylemmät kerrokset, Controller alemmat mukaanlukien radion. Laitteen osat voivat kaikki sijaita samalla mikropiirillä tai olla jaettuna eri piireille, jolloin piirien välinen kommunikaatio voi tapahtua esimerkiksi UART-väylää pitkin. [13]

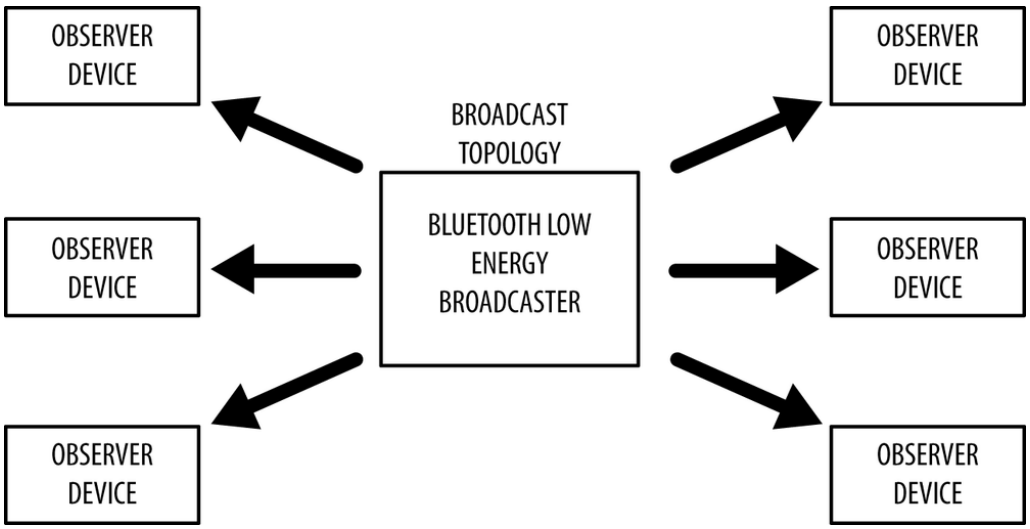
BLE on fokusoitu lyhyen kantaman kommunikointiin. Kantamaan vaikuttavat monet tekijät, kuten ympäristö ja laitteen asento. Lähetysvoimakkuutta voidaan säätää -30 ja 0 dBm:n välillä, jolloin on mahdollisa saavuttaa 30 metrin kantama ideaalisissa olosuhteissa, mutta sähkön säästämisen vuoksi tyypillinen kantama on 2 ja 5 metrin välillä. [13]



Kuva 15. *Mahdolliset Bluetooth laitetyypit [13]*

2.7.1 Verkkotopologia

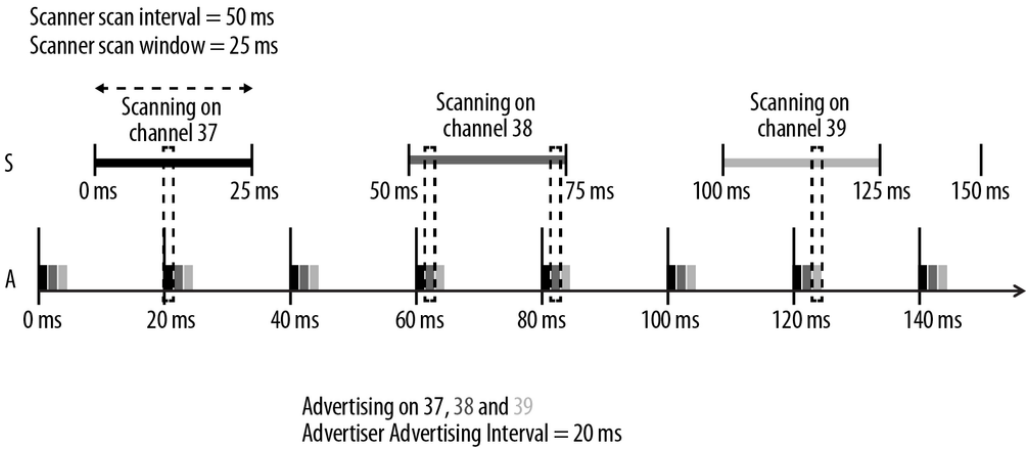
BLE-laite voi kommunikoida kahdella tavalla ympäristönsä kanssa. Vaihtoehtoja ovat lähetystopologia (engl. *Broadcast topology*) ja yhdistetty topologia (engl. *Connected topology*). Ensin mainitussa topologiassa lähettäjälaite (engl. *Broadcaster*) voi lähettää dataa mille tahansa lähialuetta tarkkailevalle laitteelle (engl. *Observer*), kuten Kuvassa 16 on esitetty. Kommunikaatio voi tapahtua tässä topologiassa vain yhteen suuntaan. Lähetystopologia on myös ainoa tapa, jolla laite voi lähettää dataa useammalle kuin yhdelle kuuntelijalle samanaikaisesti. [13]



Kuva 16. *Lähetystopologia [13]*

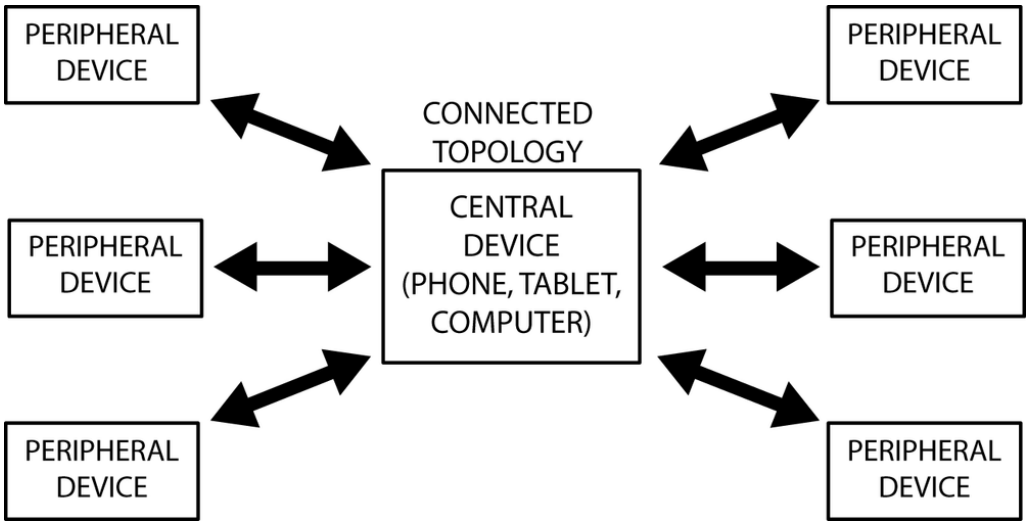
Lähetystopologiassa hyödynnetään protokollan mainostamisominaisuutta. Tavallinen mainospaketti sisältää 31 tavun hyötykuorman, joka sisältää kuvauksen lähettäjästä ja sen ominaisuuksista, mutta voi näiden sijaan sisältää mitä tahansa räätälöityä informaatiota, jota sovelluksessa halutaan käyttää. Lisäksi BLE tukee ylimääräisen mainosviestin lähetystä, mikäli data ei mahdu kokonaisuudessaan ensimmäiseen

mainosviestiin. Ylimääräisen viestin lähetys on toteutettu *Scan response* -viestinä, joka lähetetään tarkkailevan laitteen pyynnöstä. Näiden kahden mainosviestin lähetyksellä saavutetaan 62 tavun kokonaishyötykuorma. [13]



Kuva 17. *Mainostaminen ja skannaus [13]*

Mainostava laite lähettää datapaketteja sokeasti ympärilleen ilman tietoa lähellä olevasta skannaavasta laitteesta. Viestit lähetetään kiinteällä tiheydellä, joka määritetään 20 millisekunnista 10.24 sekunnin väliin sijoittuvalla mainostusintervallilla. Lyhyempi intervalli tarkoittaa tiheämpää lähetystaajuutta, mikä johtaa korkeampaan tehonkulutukseen mutta toisaalta suurempaan todennäköisyyteen siitä, että skannaava laite kykenee vastaanottammaan viestin. Mainostus tapahtuu kolmella taajuuskaistalla ja sitä ei ole synkronoitu vastaanottajan kanssa mitenkään, joten viestin vastaanotto tapahtuu satunnaisesti vastaanottavan laitteen skannausikkunan (engl. *scan window*) ajoittuessa mainosviestin kanssa päällekkäin. Kyseistä tilannetta on havainnollistettu Kuvassa 17.



Kuva 18. *Yhdistetty topologia [13]*

Mikäli dataa halutaan siirtää kahteen suuntaan tai kaksi mainosviestiä ei riitä kaiken datan siirtämiseen, täytyy laitteiden välille muodostaa yhteys. Yhteys on pysysvä tila,

jossa säännöllisin väliajoin kahden laitteen välillä tapahtuu datan vaihtoa. Yhteys on yksityinen, joten muut laitteet eivät voi vastaanottaa dataa, elleivät ne yritä laittomasti poimia viestejä ilmasta. Yhteydenmuodostukseen osallistuvat laitteet toimivat Kuvan 18 mukaisessa kahdessa roolissa:

- Keskilaite (engl. *Central*) skannaa toistuvasti lähiympäristön mainosviestejä ja havaitessaan sopivan aloittaa yhteydenmuodostuksen. Yhteydenmuodostuksen onnistuttua keskilaite hallinnoi ajoituksia ja tekee aloitteet jaksollisiin datanvaihdoksiin. Keskilaitteesta voidaan käyttää myös termiä *master*.
- Oheislaite (engl. *Peripheral* tai *Slave*) lähettää mainosviestejä säännöllisin väliajoin ja hyväksyy yhdistyspyyntöjä. Yhteyden muodostuttua oheislaite vaihtaa dataa keskilaitteen kanssa seuraten sen ajoituksia. [13]

Yhteyden muodostamiseksi keskilaite tarvitsee tiedon oheislaitteen läsnäolosta. Keskilaite etsii oheislaitteita, jotka hyväksyvät yhteydenottopyyntöjä. Mainosviestejä voidaan suodattaa laitteiden Bluetooth-laiteosoitteiden (engl. *Bluetooth device address*) perusteella, jotka vastaavat Ethernet-yhteyksissä käytettäviä MAC-osoitteista, tai mainosviestien datan perusteella. Mikäli keskilaite onnistuu vastaanottamaan mainosviestin kiinnostavalta laitteelta, se lähettää oheislaitteelle yhteydenmuodostuspyynnön, johon oheislaitteen vastatessa muodostuu yhteys. Yhteyden muodostuttua oheislaite lopettaa mainosviestien lähettämisen, joten muut keskilaitteet eivät voi tämän jälkeen havaita oheislaitetta. [13]

Yhteys laitteiden välillä tarkoittaa käytännössä jaksollista datanvaihtoa laitteiden kesken tiettyinä ajankohtina, joita kustutaan yhteystapahtumiksi (engl. *connection events*). Vaikka keskuslaite hallinnoi yhteydenmuodostusta, kumpikin laite voi vaihtaa dataa yhteystapahtuman aikana rooleista riippumatta. Huomattavaa on myös, että spesifikaation versiosta 4.1 alkaen laitteet voivat toimia samanaikaisesti sekä keskus- että oheislaitteena. Lisäksi keskuslaite voi olla yhteydessä useaan oheislaitteeseen ja oheislaite voi olla yhteydessä useaan keskuslaitteeseen. [13]

Keskeinen hyöty lähetystopologiaan nähden on siinä, että data voidaan organisoida hienojakoisesti Geneerisen Attribuutti Profiilin (GATT) mukaisesti. Tämä vähentää tiedonsiirtoa, sillä voidaan tarkemmin määrittää mikä osa datasta vaihdetaan yhteyden yli: Oheislaitteen ei tarvitse lähettää kaikkea dataa jokaisessa mainoksessa, koska se ei tiedä, kuunteleeko kukaan. Lisäksi dataa voidaan lähettää vain kun siinä on tapahtunut muutoksia. [13]

2.7.2 Geneeriset profiilit

Profiileilla määritetään BLE:n yhteydessä vastaavia datakokonaisuuksia kuin LwM2M-protokollan resurssimallissa. Profiilit sisältävät siis tietoa laitteesta ja sen antureista järjestettyinä loogisiin kokonaisuuksiin. Spesifikaatiossa määritetään

geneerisiä profiileja, jotka ovat välttämättömiä eri valmistajien laitteiden yhteensopivuuden kannalta. [13]

Geneerinen käsiksipääsyprofiili (engl. *Generic Access Profile* GAP) on pakollinen osa jokaista BLE-laitetta. Se määrittää edellä käsitellyt laitteiden roolit: keskilaitteen ja oheislaiteen sekä lisäksi kaikki proseduurit ja tilat, jotka mahdollistavat datan lähetyksen, laitteiden löytämisen yhteyksien muodostamisen ja -hallinnan. GAP on protokollan korkein hallinnointikerros. [13]

Geneerinen Attribuuttiprofiili (engl. *Generic Attribute Profile* GATT) määrittää resurssimallin ja toiminnot, jotka mahdollistavat datan lukemisen ja kirjoittamisen. GATT toimii viitekehysreferenssinä kaikille profiileille, joita laitevalmistajat voivat käyttää varmistaakseen yhteensopivuuden muiden laitevalmistajien kanssa. GATT onkin sovelluskehittäjän kannalta avainasemassa, sillä kaikki merkityksellinen data pitää jäsentää ja lähettää GATT:n sääntöjen mukaan. [13]

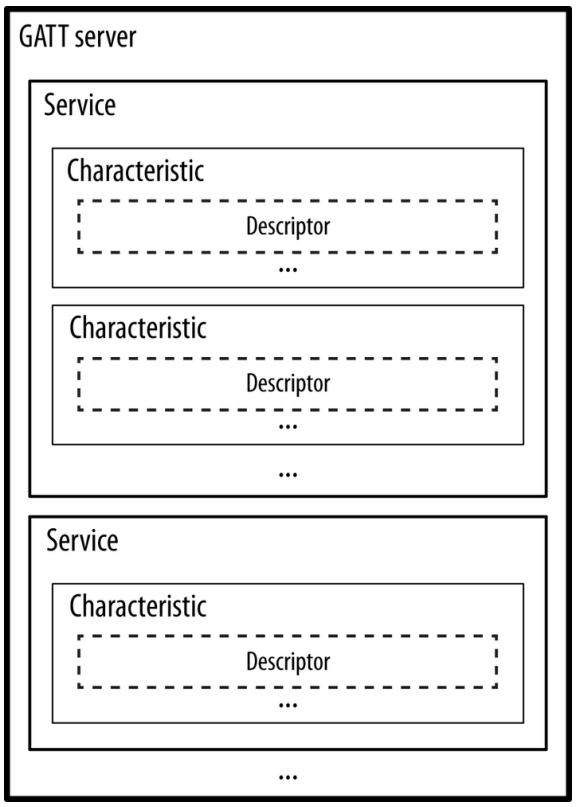
GATT käyttää siirtoporotokollanaan Atribuuttiprotokollaa (engl. *Attribute protocoll* ATT), joka määrittää jokaiselle datan osalle tunnisteen (engl. *universally unique identifier* UUID), jolla kyseiseen atribuuttiin voidaan viitata. GATT määrittää, miten atribuutit järjestetään hierarkisesti osiin, joita nimitetään palveluiksi (engl. *services*), ja palvelut jaetaan edelleen osajoukkoihin, joita nimetään ominaisuuksiksi (enlg. *characteristics*). [13]

GATT määrittää koommunikaatio-osapuolille roolit. GATT-asiakas on osapuoli, joka haluaa kerätä dataa palvelimelta. Asiakas ei tiedä palvelimen atribuutteja etukäteen, joten se joutuu ensin suorittamaan palvelun paljastuspyynnön. Atribuuttien selvittyä asiakas voi alkaa lukea ja kirjoittaa palvelimella sijaitseviin atribuutteihin. GATT-palvelin on vastuussa datan organisoinnista palveluiksi, se vastaanottaa asiakkaalta pyyntöjä ja vastaa niihin. Palvelin lähettää myös oma-aloitteisesti päivitysviestejä atribuutin tilasta, mikäli se on konfiguroitu tekemään niin, vastaavasti kuin LwM2M-kommunikaatiossa asiakasohjelma lähettää ilmoitusviestejä datan tilan muutoksista. GATT-asiakas ja palvelin ovat riippumattomia keskilaite- ja oheislaite-rooleista, eli ne voivat sijaita kummassa osapuolessa tahansa. [13]

GATT määrittää palvelimessa sijaitseville attribuuteille Kuvan 19 mukaisen hiearkian. Atribuutit on ryhmitelty palveluiksi, jotka sisältävät 0 tai useamman ominaisuuden. Ominaisuudet voivat edelleen sisältää 0 tai useamman kuvaavan elementin (engl. *descriptor*). Mikäli laite haluaa olla GATT-yhteensopiva, on sen kaikki atribuutit järjestettävä näin. Huomioitavaa on erottaa GATT-hierarkian määritys (engl. *definition*), jolla tarkoitetaan koko atribuuttijoukkoa arvoineen, ja esittely (engl. *declaration*), joka on yksittäinen atribuutti. Esittely on aina luettavissa, ja sen perusteella asiakasohjelma saa tietoa hierarkian rakenteesta. [13]

Esimerkkinä palvelimen sisällä olevasta palvelusta voidaan käyttää ”sydämen syke” -palvelua, jonka tehtävänä on mahdollistaa käyttäjän sykkeen seuranta. Palvelun esittely

yksilöi, että kyseessä on nimenomaan syke-palvelu eikä jokin muu palvelu. Palvelu sisältää kaksi ominaisuutta, jotka ovat mittauksen arvo ja kohta kehossa. Lisäksi mittauksen arvo sisältää edelleen yhden kuvaavan elementin, joka kertoo, onko ilmoitusten lähetys otettu käyttöön. Ilmoitusten lähetysten käyttöönotto määrittää sen, lähettääkö sykemittari automaattisesti mittausarvoja asiakasohjelmalle vai ei. [13]



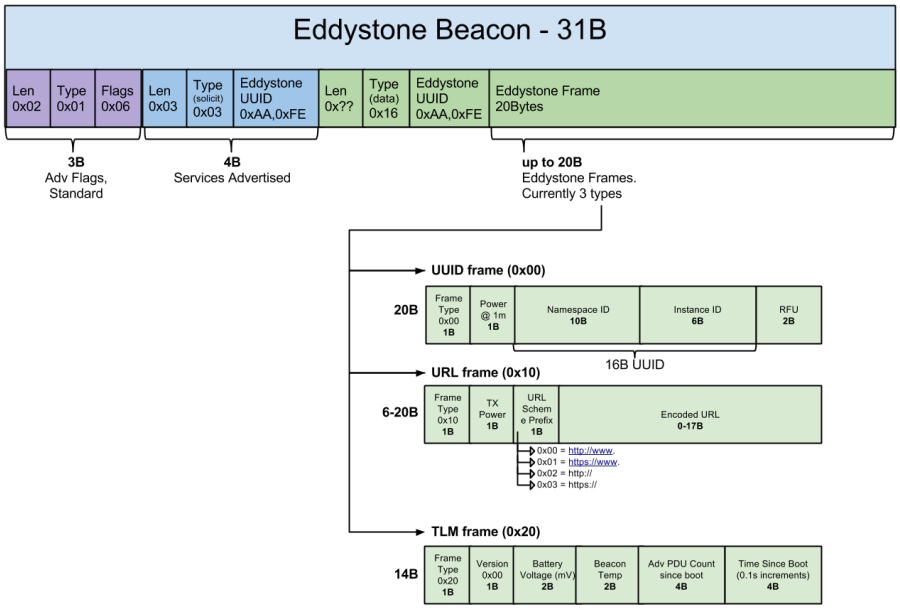
Kuva 19. *GATT-datan hierarkia [13]*

2.7.3 BLE Majakat

Laitteita, jotka käyttävät datan siirtämiseen lähetystopologiaa eli sisältävät kaiken lähetettävän informaation mainosviesteihin, kutsutaan majakoiksi (engl. *beacon*). Google on kehittänyt Eddystone-nimisen avoimen lähdekoodin majakkaprotokollan, jota voidaan käyttää muun muassa sisätilapaikannukseen ja ilmoitusten lähettämiseen lähellä oleviin matkapuhelimiin. Eddystone tarjoaa 4 erilaista datakehystä, joita protokollaa käyttävä BLE-mainostaja voi lähettää [25]:

- Eddystone UID -kehys sisältää 16 bittiä pitkän tunnisteen, jota voidaan käyttää laukaisemaan jokin toiminto sovelluksessa.
- Eddystone URL on kehys, johon voi sisällyttää pakatun URL-osoitteen, josta sovellus voi internetin kautta hakea lisää informaatiota. URL-kehyksellä voidaan tehdä fyysisen internetin sovelluksia, jossa käyttäjän tulee mennä tiettyyn paikkaan tietääkseen tietyn verkko-osoitteen.

- Eddystone TLM -kehys on tarkoitettu majakan anturidatan lähettämiseen. Anturidataa voi olla esimerkiksi akun jännite ja ympäristön lämpötila.
- Eddystone EID -kehys sisältää vaihtuvan salatun tunnisteen, jota on tarkoitus käyttää verkkopalveluun kirjautumisessa.

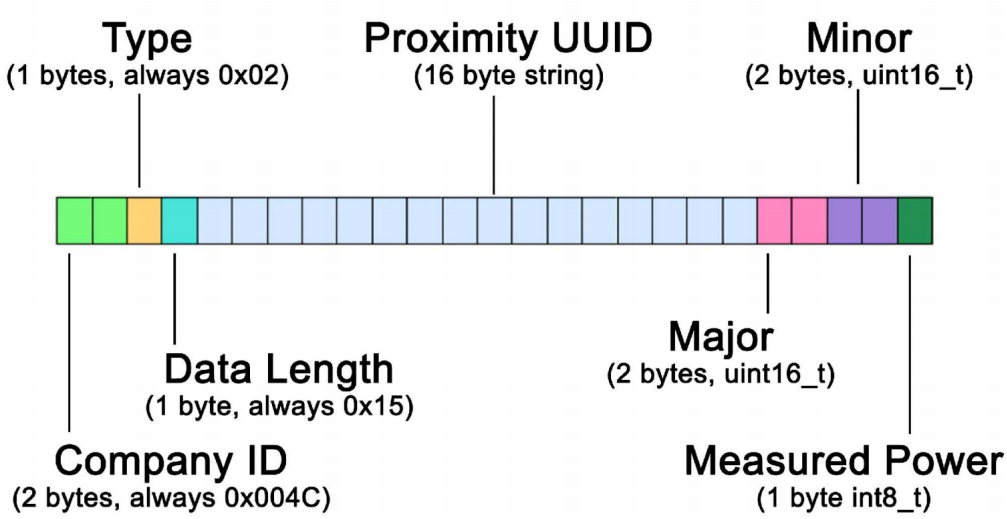


Kuva 20. *Eddystone-kehys [25]*

Eddystone-formaatin mukainen BLE-mainospaketti on esitetty Kuvassa 20. Kuvasta nähdään, että mainospaketti on 31 tavua pitkä. Kolme ensimmäistä tavua käytetään lippuihin, jotka määrittävät paketin mainosviestiksi. Seuraavat tavut määrittävät, mitä GATT-palveluita kyseisessä mainosviestissä mainostetaan. Eddystonen tapauksessa palveluita on vain yksi, Eddystone, joten tavuja tarvitaan 4. Seuraavat 4 tavua määrittävät, että mainosviesti sisältää myös dataa Eddystone-formaatissa. Varsinaiseen datakehykseen jää siis 20 tavua. Datakehyksen ensimmäinen tavu määrittää, mikä neljästä mahdollisesta kehyksestä on kyseessä, ja loput tavuista sisältävät kyseisen kehyksen datan. [25]

Ruuvi-niminen yritys valmistaa Ruuvitag-nimisiä BLE-majakoita [30]. Laitteet sisältävät ilmanlaatua mittaavia antureita sekä kiihtyvyysanturin. Ruuvitag tukee anturidatan lähetyksessä Eddystone URI -kehystä. URI-kehys sisältää verkko-osoitteen, johon on koodattu ilmanlaatuanturin arvot. URI voi olla esimerkiksi *http://ruu.vi#AjAYAMLs*, johon selaimella siirtymällä nähdään Ruuvitagin mittaamat ilman lämpötila, -kosteus ja -paine. Kyseiset luvut on koodattu kirjaimiin ”AjAYAMLs” base 64 -koodauksella. Koodauksen purkamisen jälkeen kirjaimista saadaan 7 tavua, joista ensimmäinen sisältää kosteusarvon puolen prosentin tarkkuudella. Seuraavat kaksi sisältävät lämpötilan kahden desimaalin tarkkuudella. Ilmanpaine on sisällytetty tavuihin

4 ja 5. Loput 2 tavua (0. ja 6.) sisältävät formaattimäärityksen ja satunnaisen tunnisteen, jolla ei ole datan keräyksen kannalta merkitystä. [29]



Kuva 21. *iBeacon-formaatti [17]*

Ruuvitag tukee Eddystonen lisäksi omaa raakaformaattiaan, jolla saadaan mainospakettiin enemmän dataa käyttämällä hyödyksi myös tavut, jotka käytetään URI-kehyksessä ”http://ruu.vi#” -osan muodostamiseen. Saadut lisätavut käytetään raakaformaatissa kiihtyvyysanturin mittaamien kiihtyvyyksien lähettämiseen X-, Y- ja Z-akseleiden suhteen. Myös laitteen pariston jännite lähetetään raakaformaatissa. [29]

Ibeacon on Applen kehittämämä BLE-majakkastandardi vuodelta 2013. IBeacon-majakan ainoa tarkoitus on lähettää sen tunnistetta ja radiosignaalin tehoa. Majakkaa on siis tarkoitus käyttää Eddystone UID -kehyksen tavoin paikannukseen, jossa mainosviestejä vastaanottava laite voi arvioida etäisyyttä lähettävään majakkaan. Tunnisteen ja radiotehon lisäksi Ibeacon sisältää 2 tavun Major- ja 2 tavun Minor-kentät, joita voidaan käyttää datan lähetykseen. Tyypillinen iBeacon-sovellus on taidemuseon sisällä tapahtuva paikannus, jossa käyttäjän puhelimeen tulee ilmoituksena tietoisku näyttelyesineestä käyttäjän ollessa 2 metrin päässä kyseisestä taideteoksesta. [17]

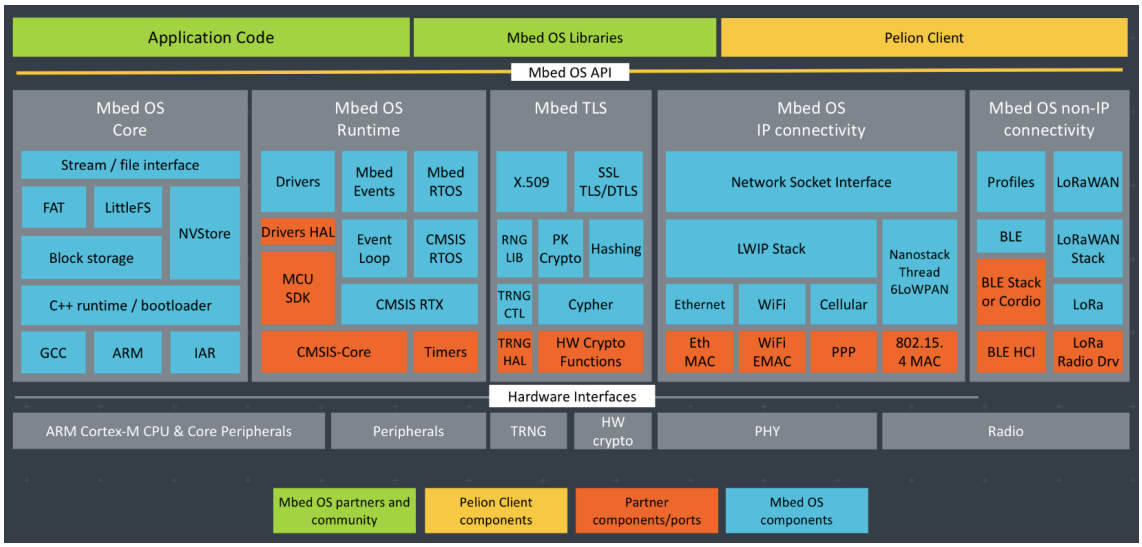
IBeacon-kehys on esitetty Kuvassa 21. Kehyksen alussa on kahden tavun mittainen yritystunniste. joka on Ibeaconin tapauksessa aina Appleen viittaava 0x004C. Seuraavaksi on mainostyyppi-kenttä, joka kertoo, että kyseessä on iBeacon-viesti. Seuraava tavu ilmoittaa datan pituuden, joka on myös vakio 0x15. Varsinaisen datan ensimmäinen kenttä on 16 tavun tunniste, jonka voi rekisteröidä Applen rekisteriin. Tunniste kertoo, mistä laitteesta on kyse. Tunnisteen pituus varmistaa, ettei kahden eri valmistajan majakoilla voi olla samaa tunnistetta. Tunnisteen jälkeen on 2 kenttää, Major ja Minor, molemmat pituudeltaan 2 tavua. Kyseisiä lukuja voidaan käyttää ilmaisemaan, mikä yksittäinen laite on kyseessä. Nämä kentät ovat myös ainoat, joita voidaan käyttää anturidatan välittämiseen, joten iBeaconin mahdollisuudet lähettää anturidataa ovat huomattavasti rajallisemmat verrattaesssa Eddystoneen. Viimeinen yhden tavun kenttä

sisältää lähetystehon, ja sovellus voi käyttää sitä laskiessaan etäisyyttä kyseiseen majakkaan. [17]

Cypress valmistaa iBeacon-formaattia tukevia Cyalkit-nimisiä BLE-majakoita. Majakoissa on anturit, jotka mittaavat ilman lämpötilaa ja kosteutta. Cyalkit-majakoiden mainosviestissä rekisteröity UUID on ”00050001-0000-1000-8000-00805F9B0131”. Cyalkit-majakat käyttävät Major-kenttää ilmaisemaan, mikä majakoista on kyseessä. Laitteet myydään viiden kappaleen pakkauksissa, joten Major-kentän oletusarvo on luku väliltä 1–5. Anturidataa Cyalkit lähettää sisällyttämällä Minor-kentän ensimmäiseen tavuun ilmankosteuden ja jälkimmäiseen tavuun lämpötilan arvon. [33]

2.8 Arm Mbed

Mbed on Armin kehittämä ohjelmistokehys (engl. *programming framework*), joka tarjoaa tuen laajalle valikoimalle sulautettuja laitteita. Kuten Kuvasta 22 nähdään, Mbed tarjoaa käyttöjärjestelmän (ARM Mbed OS), pilvipalveluita (asiakas- ja palvelinkomponentteja, joilla saadaan Mbed-laitteet kommunikoimaan kesekenään), valtavan joukon kirjastoja sekä työkalut ohjelmiston kääntämiseen ja konfigurointiin. [20]



Kuva 22. *ARM Mbed OS [20]*

Mbed OS on ohjelmistokehyksen keskeisin komponentti, sillä se sisältää kaikki IoT-sovelluksen kehittämisessä tarvittavat komponentit, kuten ajurit antureille ja kommunikointiin tarvittaville modeemeille, sekä tietoturvaan tarvittavat ohjelmistomoduulit. Mbed-ekosysteemi tarjoaa internetissä selaimella toimivan kehitysympäristön (engl. *integrated development enviroment,* IDE), jolla pääsee Mbed-OS:ään käsiksi ilman esivaatimuksia. IDE mahdollistaa nopean prototypoinnin matalahintaisilla yhteistyökumppaneiden valmistamilla kehityslaudoilla. Mbed-OS:n lähdekoodi on myös avoin, joten sen voi ladata omalle koneelleen ja kääntää lokaalisti Mbed:in tarjoamalla python-komentorivityökalulla. Mbed-yhteensopivien

kehityslautojen ohjelmointi on myös helppoa, sillä ohjelmointi tapahtuu kytkemällä kehityslaitealusta tietokoneeseen USB-liitännällä ja siirtämällä käännetty ohjelma tiedostojärjestelmään ilmestyneeseen kansioon. [20]

Mbed OS käyttää raudan abstraktointikerrosta (engl. *hardware abstraction layer*, HAL) mahdollistamaan laiteriippumattoman ohjelman kirjoittamisen. Ohjelmakoodiin sisällytetään automaattisesti yleisimmät oheislaitteiden kanssa kommunikointiin tarvittavat kirjastot kuten SPI ja I2C, joiden käyttämistä varten on olemassa ohjelmointirajapinnat (engl. *application programming interface*, API). HAL mahdollistaa myös laajennukset Mbed:iä tukevien prosessorien valikoimaan. MBed OS:n ydin on käyttöjärjestelmä nieltä RTOS (engl. *Real Time Operating System*), joka tukee rinnakkainsuoritusta ja reaaliaikasuoritusta. RTOS tuo mukanaan ominaisuuksia, kuten suoritussäikeet ja muteksit. [20]

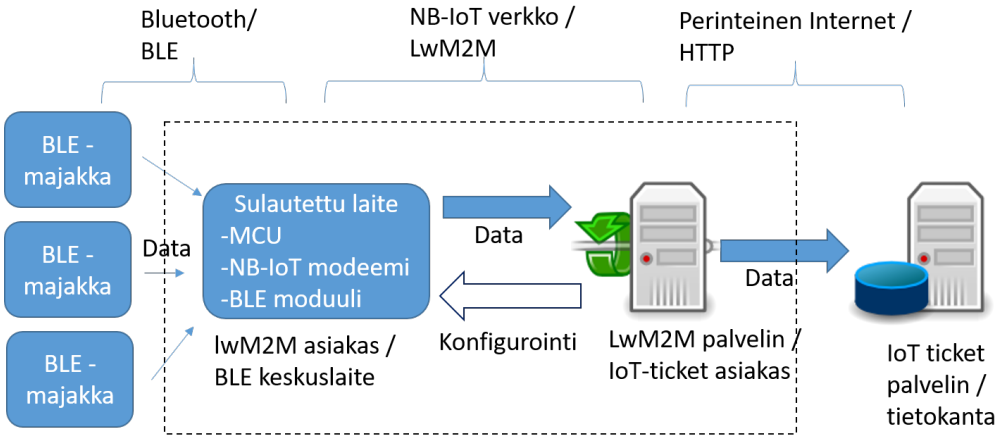
Armin ekosysteemissä on useita kommunikointiin erikoistuneita yhteistyökumppaneita, joten Mbed OS :n tukemia verkkotekniikoita on muunmuassa BLE, NFC, RFID, LoRa, Ethernet, Wi-Fi ja puhelinverkkoon perustuvat IoT-ratkaisut. Mbed OS:n tarjoamat verkkoprotokollapinot ovat joustavia, joten niillä voidaan toteuttaa yksinkertainen mikrokontrollerilla ja radiolla toimiva sovellus tai monimutkainen useaan datavälylällä toisiinsa kytkettyjen piirien sovellus. Useat Mbedin pinoista on hyvin sertifioituja ja testattuja, joten niiden pitäisi olla myös luotettavia. Kommunikointiin liittyen Mbed tarjoaa myös TLS-salauksen ja kehityksen alla olevan tuen laiteohjelmiston etäpäivitykseen. [20]

Edellä mainittujen ominaisuuksien lisäksi Mbed OS:n valintaan liittyi erityisesti siinä kehityksen alla ollut tuki Quectelin BG96 NB-IoT -modeemille. Toinen merkittävä tekijä oli Mbed Client -nimellä kulkeva LwM2M asiakasohjelman toteuttamiseen tarkoitettu kirjasto. Kirjaston käytöstä on esimerkkisovellus *mbed-os-example-client*, jonka pohjalta on helppo rakentaa oma LwM2M-asiakasohjelma. Esimerkkiohjelma ottaa yhteyttä Mbedin omaan pilvipalveluun, josta laitteen voi komentaa vilkuttamaan ledejä pilvestä erikseen asetettavalla vilkutuskuviolla. Lisäksi pilvipalvelusta voi lukea, kuinka monesti laitteessa olevaa nappia on painettu. [19]

3. TYÖN SUORITUS

Sulautettuja laitteita suunniteleva ja valmistava Wapice Oy haluaa tutkia uuden NB-IoT-verkon mahdollisuuksia tulevaisuuden tuotekehityksessä. Yritys antoi työn aiheeksi luoda kyseistä verkkotekniikkaa käyttävän proof-of-concept-laitteen messukäyttöä varten. Työn edetessä laitteeksi sovittiin niin sanottu BLE-NB-IoT Gateway, eli laite, jonka kautta useampi antureita sisältävä BLE-laite voi lähettää dataa Wapicen omaan IoT-Ticket-pilvipalveluun. Keskeiseksi ratkaistavaksi onglemaksi työssä muodostui se, miten pienen datasiirtonopeuden omaavassa NB-IoT-verkossa saataisiin siirrettyä dataa palvelimelle, jossa datan vastaanottamiseen oli tarjolla vain raskas REST-rajapinta. Laitteen konfiguroitavuus etänä oli myös yksi halutuista ominaisuuksista, mikä aiheutti toisen ratkaistavan ongelman.

Kuvassa 23 on esitetty kokonaiskuva ratkaisusta, jolla edellä esitettyihin ongelmiin pureuduttiin. Kuvassa vasemalla on BLE-majakoita, joiden on tarkoitus toimia anturidatan lähteenä ja lähettää sitä ympärilleen BLE-mainosviesteinä. Työssä suunniteltavan sulautetun IoT-laitteen on tarkoitus vastaanottaa mainosviestit, eli toimia BLE protokollan keskuslaitteena, ja lähettää kerätty data eteenpäin NB-IoT-verkossa. Tietoliikenneverkon ylemmän kerroksen protokollaksi valittiin LwM2M, koska se on HTTP-protokollaa kevyempi ja tarjoaa viestinlähetysmahdollisuuden palvelimelta asiakkaalle mahdollistaen asiakkaan etäkonfiguroitavuuden.



Kuva 23. *Yleiskuva suunnitellusta ratkaisusta*

IoT-Ticketistä puuttuvan LwM2M-tuen vuoksi jouduttiin päättämään, tehdäänkö tuki suoraan siihen vai tehdäänkö erillinen LwM2M-palvelin, joka lähettää vastaanottamansa datan edelleen IoT-Ticketin tarjoamaan REST-rajapintaan [36]. Työssä päädyttiin

jälkimmäiseen, koska se osoittautui helpommaksi toteuttaa. Mahdollinen integraatio jätettiin työn rajauksen ulkopuolelle. Kuvasta 23 nähdään, miten datan on tarkoitus siirtyä Nb-IoT-verkossa ensin LwM2M-asiakkaalta palvelimelle ja jatkaa tämän jälkeen HTTP-protokollan mukaisesti IoT-Ticketiin. Tarkkaan ottaen data kulkee NB-IoT-verkossa ainoastaan operaattorin tukiasemalle ja vaihtaa siinä toiseen siirtotapaan, mutta se ei ole työn kannalta olennaista. Datan lisäksi LwM2M-palvelimelta voidaan lähettää konfigurointiviestejä myös asiakkaalle, mikä on esitetty Kuvassa 23 valkoisella nuolella.

Tässä luvussa kuvataan tarkemmin työn vaiheita. Ensin kerrotaan, mistä HW-komponenteista laitteisto rakennettiin ja miksi niihin päädyttiin. Sen jälkeen kuvataan ohjelmistoarkkitehtuuri ensin laitteen ja sitten palvelimen osalta.

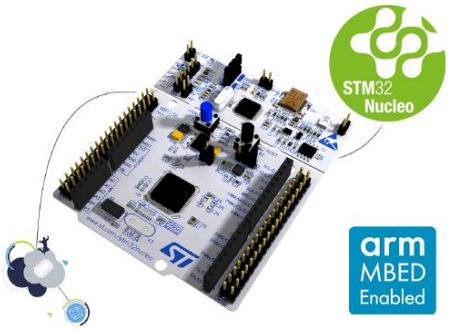
3.1 Laitteisto

Työssä oli tarkoitus tutustua itse verkkotekniikan lisäksi Mbed-OS:ään ja sen mahdollisuuksiin prototyyppien kehittelemisessä, joten ensisijaisesti oli selvitettävä, voisiko halutun laitteen toteuttaa Mbed-OS:ää hyödyntäen. Mbed-OS listaa verkkosivullaan kaikki sen tukemat kehityslaudat, joissa ominaisuudet vaihtelevat muistin ja suorituskyvyn suhteen. Lähtökohtaisesti NB-IoT on suunnattu yksinkertaisten ja pienivirtakulutuksisten laitteiden kommunikaatioväyläksi, joten tuotantoa ajatellen laitteiden yksikköhinta on tärkeä ja siksi laitteistoksi sopisi joku halvemman prosessorin sisältävä kehitysalusta.

Laitteiston hintaa suuremmaksi ongelmaksi muodostui kuitenkin NB-IoT-verkon uutuudesta johtuva markkinoilla olevien modeemien vähyys. Modeemeja tarjosivat projektin alkuvaiheessa ainoastaan UBLOX [34] ja Quectel [26]. Modeemeille ei myöskään ollut Mbed-OS:n tukemaa valmista kirjastoa, joskin niille on myöhemmin sellainen kirjoitettu, joten laitteiston valinnasta oli tulossa haastavaa. Avnet Silica -niminen yritys oli kuitenkin tehnyt Quectelin BG96 -modeemin sisältävän niin sanotun laajennusalustan (engl. *shield*) eli piirilevyn, joka voidaan kytkeä kehityslaitealustaan siinä olevien kytkentärimojen kautta. Lisäksi yritys oli tehnyt demo-ohjelman [32], jossa MBed-yhteensopiva kehityslaitealusta oli saatu lähettämään dataa NB-IoT-verkossa kyseisen laajennusalustan avustamana.

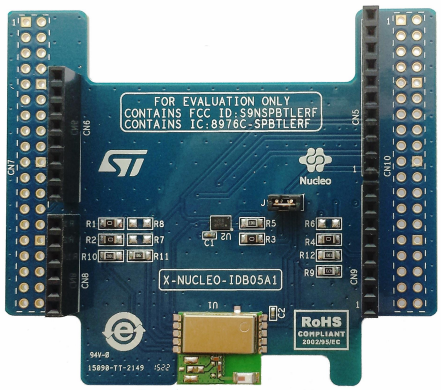
Prototyyppiin valittiin Avnet-demon innoittamana kyseisessä demossa käytetty ST-microelectronicsin Nucleo L476RG -kehityslaitealusta (engl. *development board*) [21], joka on esitetty Kuvassa 24 (a), ja Avnet silican -laajennusalusta, joka on esitetty Kuvassa 24 (b). ST-microeletroniksin-kehityslaitealusta sisältää ARM:n 32-bittisen M4-sarjan mikrokontollerin. Mikrokontrolleri toimii 80 MHz:n maksimikellotaajuudella, ja siinä on 1 MB Flash-muistia sekä 128 kB SRAM-käyttömuistia. Työn kannalta muita olennaisia ominaisuuksia on UART, jota mikrokontrolleri käyttää NB-IoT-modeemin kanssa kommunikointiin, satunnaislukugeneraattori, jota viestinnän salausalgoritmit

tarvitsevat, SPI, jolla prosessori voi kommunikoida BLE-modeemin kanssa, sekä ajastimet, joita prosessori käyttää ajoittamaan viestimistä.



(a) *Nucleo-kehityslaitealusta [21]*

(b) *Nb-IoT-laajennusalusta [32]*



(c) *BLE-laajennusalusta [23]*

Kuva 24. *Laitteistoon käytetyt piirilevyt*

Datan lähetysmekanismin lisäksi tarvittiin datan lähde, jotta laite olisi ylipäätään mielekäs rakentaa. NB-IoT-verkko on suunnattu pienen datamäärän lähetykseen, joten tyypillinen datan lähde tällaisessa sovelluksessa olisi voinut olla jokin ympäristöä mittaava anturi, kuten lämpömittari. Tässä työssä haluttiin kuitenkin tehdä geneerisempi BLE-yhteyspiste, jossa datan lähde on kolmannen osapuolen laite, joka lähettää oman anturidatansa BLE:n välityksellä työssä suunniteltavalle laitteelle, joka edelleenlähettää saamansa datan internetiin. BLE-viestimistä varten hankittiin St-microelectronicsin laajennusalusta [23], joka on esitetty Kuvassa 24 (c). Valinta johtui pääosin siitä, että kyseiselle laajennusalustalle oli valmiina Mbedin tarjoama ohjelmistokirjasto, mikäli kyseistä laajennusalustaa käytettäisin jo hankitun Nucleo-kehityslaudan kanssa.

Osa Bluetooth-laajennusalustan ja NB-IoT-laajennusalustan rimaliittimien kytkentänastoista oli molempien käytössä, mikä tarkoittaa sitä, että laajennusalustoja ei sellaisenaan voinut käyttää yhtä aikaa. Osa toisen laajennusalustan nastoista tuli katkaista, jotta ne eivät muodosta galvaanista yhteyttä toisen laajennusalustan kanssa, ja uudelleenkytkeä hyppylankojen avulla toisista nastoista. Kehityslaitealusta ja laajennusalustat ladottiin siten, että alimmaisena oli itse kehityslaitealusta, keskimmäisenä BLE-laajennusalusta ja päällimmäisenä NB-IoT-laajennusalusta, jonka

pinneistä osa katkaistiin ja uudelleenkytkettiin. Yhteisessä käytössä olivat alunperin kehityslautojen kytkentänastat D7 ja D11. D7 oli molemmilla käytössä reset-kytkentänastana ja resetin polariteetti oli sama, joten sille ei tarvinnut tehdä mitään. D11-kytkentänasta on kuitenkin BLE-laajennusalustan SPI-väylän käytössä. NB-IoT-laajennusalusta käytti sitä taas päällekytkemiseen, joten kyseinen kytkentänasta piti katkaista NB-IoT-laajennusalustan rimaliittimestä ja kytkeä uudelleen hyppylangalla toisesta kytkentänastasta. Lisäksi BLE-laajennusalusta tarvitsee nastat D3, D12, A0 ja A1 SPI-kommunikaatioon. Vaikka kyseiset nastat eivät olleet NB-IoT-laajennusalustan käytössä ainakaan ohjelmakirjastojen perusteella, ne kuitenkin häiritsivät BLE-laitteen toimintaa, joten myös kyseiset nastat katkaistiin kytkemättä niitä muualta hyppylangalla.

(a) *Ruuvitag [30]* ***(b)*** *Cyalkit-E03 [12]*

Kuva 25. *Datan lähteinä käytetyt BLE-majakat*

Datan lähteeksi valittiin kahden eri valmistajan BLE-majakoita. Toinen majakoista on suomalaisen Ruuvi Oy:n valmistama Ruuvitag [30], joka on esitetty Kuvassa 25 (a). Ruuvitag sisältää 3 anturia, joilla laite mittaa ilman lämpötilaa, -kosteutta ja -painetta. Lisäksi laitteessa on kiihtyvyysanturi, joka mittaa kiihtyvyyttä 3 akselin suhteen (X, Y ja Z). Laite käyttää virtalähteenään paristoa, ja laitteelle luvataan 4 vuoden käyttöaika laitteen tehdasasetuksilla. Majakka tukee 2 eri protokollaa, jolla se voi sisällyttää mainosviesteihinsä omien antureidensa mittausarvot. Toinen on Googlen Eddystone -protokolla, jonka URI-formaattiin on koodattu ilmaa tarkkailevien antureiden arvot. Toinen on Ruuvin oma raakaformaatti, joka sisältää myös kiihtyvyysanturin arvot.

Toisena majakkana käytettiin Cypress Energyn valmistamaa Cyalkit-E03:a [12], joka on esitetty Kuvassa 25 (b). Cyalkitillä on anturit ainoastaan ilman lämpötilan ja kosteuden mittaamiseen, ja laitteet käyttävät energianlähteenään aurinkopaneelia. Majakat tukevat Applen iBeacon-protokollaa. Laitteet koodaavat anturiensa lukemat iBeacon-protokollan sisälle Minor-kenttään.

3.2 Ohjelmisto

Ohjelmiston tavoitteena oli lähettää dataa edellisessä osuudessa kuvatulta laitteelta Wapicen pilvipalveluun nimeltä IoT-Ticket. Datan lähetysprotokollana haluttiin käyttää LwM2M-protokollaa, koska sen käyttämä CoAP kapseloi saman informaation pienempään kokonaisdataan kuin HTTP. Lisäksi LwM2M tarjoaa mahdollisuuden hallita laitetta muuttamalla sen asetuksia esimerkiksi yhteydenottojen välisten unijaksojen pituuden suhteen.

IoT-Ticket ei tarjoa valmiiksi LwM2M-serverirajapintaa, vaan siinä datan vastaanottoon on tarjolla ainoastaan Wapicen oma REST-rajapinta [36], johon viestejä voidaan lähettää HTTP-yhteyden kautta. LwM2M-palvelinta ei myöskään haluttu integroida suoraan osaksi IoT-Ticketiä. Tämän vuoksi päädyttiin ratkaisuun, jossa erilliselle palvelimelle ohjelmoitiin LwM2M-palvelinosuus, jonka kanssa laitteen tuli suorittaa viestien vaihtaminen. Saman palvelimen tuli tarjota verkkosivu, jonka kautta käyttäjä pystyy hallitsemaan laitetta. Palvelimen tuli myös lähettää uudelleen laitteelta saatu data IoT-Ticketiin.

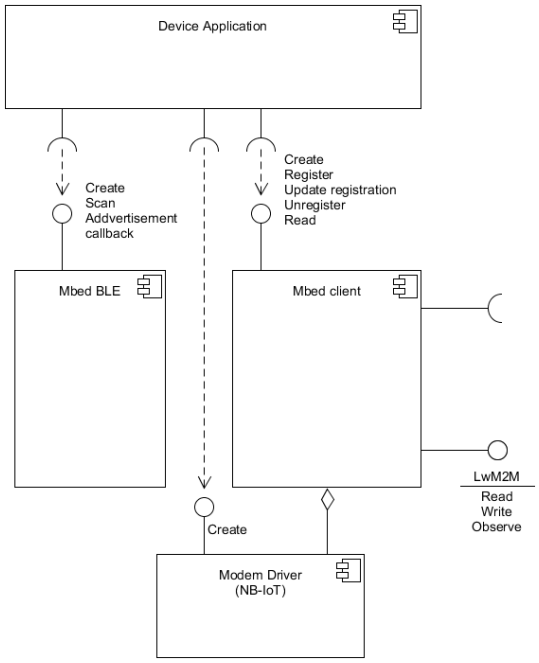
Kokonaiskuva ohjelmistosta on esitetty Liitteessä A. Ohjelmisto voidaan jakaa karkeasti 2 osaan, joista ensimmäinen on IoT-laitteeseen ja toinen palvelimelle ohjelmoitu osa. Laitteen ohjelmisto jakaantuu BLE-kommunikaatiosta vastaavaan ja palvelimen kanssa kommunikoivaan osaan. Palvelimen ohjelma voidaan vastaavasti jakaa kahtia laitteen kanssa kommunikoivaan osaan ja IoT-Ticketin kanssa kommunikointiin. Lisäksi palvelin tarjoaa selainkäyttöliittymän, jonka kautta käyttäjä voi hallita laitteen asetuksia ja päättää, mitä dataa palvelin edelleenlähettää IoT-Ticketiin. Tässä luvussa kuvataan tarkemmin, miten laitteen ja palvelimen ohjelmistot rakentuivat.

3.2.1 Sulautetun laitteen ohjelmisto

Sulautetun laitteen osa kokonaisarkkitehtuurista on esitetty Kuvan 26 UML-kaaviossa. Laitteen tarkoitus kokonaissysteemissä on kerätä sen fyysisessä läheisyydessä olevilta BLE-majakoilta sensoridataa ja lähettää se internetin välityksellä LwM2M-palvelimelle. Laitteen ohjelmisto tehtiin Mbed-OS:n tarjoamien kirjastojen pohjalta. MBed-OS on toteutettu ohjelmointikielellä C++, joten kaikki sen tarjoamat kirjastorajapinnat on toteutettu luokkina. Arkkitehtuurin UML-kaaviosta voidaan havaita, että sovellus käyttää BLE-toiminnallisuuden toteuttamiseen Mbed BLE -luokkaa, joka tarjoaa rajapinnan BLE-mainosviestien skannaamiseen ja Callback-funktion vastaanotettujen mainosviestien käsittelyyn.

LwM2M–asiakas-viestintään internetissä Mbed-OS tarjoaa Mbed-Client-luokan, jonka rajapinta tarjoaa funktiot palvelimelle rekisteröitymiseen ja rekisteröinnin uusimiseen, mikä on vaadittu LwM2M-protokollassa. Lisäksi luokka tarjoaa mahdollisuuden datan lähettämiseen ja vastaanottamiseen LwM2M-palvelimelta protokollaan liittyvien objektien avulla. MBed-client ei tarjoa suoraan funktioita datan lähetykseen, vaan

rajapinnan LwM2M-objektien luomiseen, joista lähetetään palvelimelle lista rekisteröinnin tai sen uusimisen yhteydessä. Kun sovelluksessa tehdään muutos johonkin kyseisessä listassa olevista objekteista, lähettää Mbed-client automaattisesti päivityksen muutoksesta palvelimelle, mikäli palvelin on tätä ennen lähettänyt Observe-pyynnön kyseiseen resurssiin. Vastaavasti datan vastaanottaminen palvelimelta aiheuttaa Callback-funktion suorituksen, jonka perusteella sovellus saa tiedon vastaanotetusta datasta. Mbed-client on geneerinen toteutus, joka ei riipu käytettävästä verkosta. Tämän vuoksi sovelluksen on alustettava sille myös verkkorajapinta, joka kommunikoi tässä työssä käytettävällä Nb-IoT-modeemilla.



Kuva 26. *Sulautetun laitteen arkkitehtuuri*

Edellä mainittuja luokkia käyttäen laitteen ohjelma toimii siten, että ensin se alustaa internetyhteyden, sitten luo kaikki LwM2M-resurssit sekä datan lähetykseen että laitteen konfiguraation säilyttämiseen, ja lopuksi rekisteröi kyseiset resurssit LwM2M-palvelimelle. Tämän jälkeen ohjelma siirtyy ikuiseen silmukkaan. Silmukan alussa laite lukee konfiguroinnistaan skannausajan, jonka palvelin on saattanut muuttaa edellisen lukukerran jälkeen, ja aloittaa kyseisen ajan kestävän BLE-skannauksen. Skannauksen aikana vastaanotetut mainosviestit joko tallennetaan edelleenlähetettäviksi tai hylätään riippuen siitä, onko palvelin konfiguruinut niiden MAC-osoitteet mielenkiintoisiksi vai ei. Skannauksen jälkeen sovellus kirjoittaa edelleenlähetettäviksi määritetyt viestit mainosviesteille varattuun LwM2M-resurssiin yksitellen, ja Mbed-Client-olio aiheuttaa automaatisen dataviestin kyseisistä resurssimuutoksista. Resurssiin kirjoittamisen jälkeen sovellus lukee konfiguroinnistaan idle-ajan, jonka aikana ohjelma ei tee mitään verkkoliikenteen vähentämiseksi ja siten myös sähkön säästämiseksi. Idle-ajan päätyttyä laite aloittaa saman sekvenssin uudelleen.

```
#include "mbed.h"
#include "easy-connect/easy-connect.h"

MbedClient mbed_client();
int main() {
    //Luodaan verkkorajapinta olio
    NetworkInterface* network = easy_connect(true);
    ...

    // Luodaan LwM2M asiakasrajapintaolio
    mbed_client.create_interface(MBED_SERVER_ADDRESS,
        network);

    //luodaan LwM2M objekti, joka sisaltaa rekisterointiin
    //tarvittavat tiedot
    M2MSecurity* register_object =
        mbed_client.create_register_object();

    // Luodaan laiteobjekti, joka on pakollinen LwM2M
    // standardissa
    M2MDevice*   device_object   =
        mbed_client.create_device_object();

    // luodaan lista joka sisaltaa kaikki objektit
    // security-objektia lukuunottamatta
    M2MObjectList object_list;

    // Add lisataan olaiteobjekti listaan
    object_list.push_back(device_object);
    // (lisataan mahdolliset muut objektit listaan)
    ...

    // asetetaan luotu rekisterointiolio
    mbed_client.set_register_object(register_object);

    // rekisteroidaan lwM2M client objekteineen
    // palvelimelle
    mbed_client.test_register(register_object,
        object_list);
    while (true) {
        ...
    }
}
```

Ohjelma 1. *Internet-rajapinnan ja LwM2M-asiakkaan luominen Mbedin kirjastoilla*

Mbed-OS:n kirjaston easy-connect [4] tarjoaa internetyhteyden muodostukseen yleiskäyttöisen *easy_connect()*-funktion, jonka paluuarvona kutsuja saa internetrajapinta-olion, kuten on esitetty Ohjelmassa 1. Mikäli tuettu kehityslaitealusta (tai laajennusalusta) sisältää jonkin verkkolaitteen, kuten ethernet-liittimen, Wi-Fi-piirin tai tämän työn tapauksessa NB-IoT-modeemin, pitäisi funktiokutsun riittää internetyhteyden muodostamiseen, ja paluuarvona saatava olio vastaisi käytettävää internet-rajapintaa. Jotta main.cpp-tiedosto toimii kaikilla rajapinnoilla, tulee Mbed-OS-projektiin sisällyttää *mbed_app.json*-tiedosto. Kyseiseen JSON-tiedostoon tulee kirjoittaa, mitä verkkoajuria halutaan käyttää. Käytännössä JSON-tiedostosta generoidaan joukko DEFINE-makroja, jotka kertovat varsinaiselle kääntäjälle, mitkä osat Mbed-OS:n kirjastoista käännetään mukaan varsinaiseen suoritettavaan ohjelmaan.

```
{
    "macros": [
        "CELLULAR_DEVICE=QUECTEL_BG96",
        "MDMTXD=D8",
        "MDMRXD=D2"
    ],
    "target_overrides": {
        "*": {
            "target.features_add": [ "COMMON_PAL"],
            "platform.default-serial-baud-rate": 115200
        },
        "NUCLEO_L476RG": {
            "target.features_add": ["BLE"],
            "target.extra_labels_add": ["ST_BLUENRG"]
        }
    }

}
```

Ohjelma 2. *MBed-OS:n konfigurointi mbed_app.json-tiedostolla*

Mbed-OS (versio 5.8) ei suoraan tukenut käytettävää laajennusalustaa tarjoamalla sille valmista konfigurointia JSON-tiedostoon. Mbed:ssä kuitenkin oli kehityksen alainen ajuri käyettävälle modeemille, ja Avnet tarjoaa laajennusalustaa käyttävän esimerkkiohjelman [8], joten shiedin käyttö oli mahdollista. Koska laajennusalustan ajurille ei ollut valmista JSON-konfiguraatiota, Mbed-OS:n kirjastoista piti tutkia, mitä makroja pitää konfiguroida, jotta *easy_connect()*-funktiota voitiin käyttää. Kyseinen funktio voidaan konfiguroida palauttamaan EasyCellularConnection-olio [2], jonka rakentajan parameteihin kuuluu makroina modeemin käyttämät sarjayhteyden TX- ja RX-kytkentänastat sekä sarjayhteyden baudinopeus. Lisäksi EasyCellularConnection-olion määrittelytiedostossa vaaditaan makro käytettävästä ajurista, joka kapseloi modeemin käyttöön tarvittavat AT-komennot.

Halutut makrot Mbed-OS:n kofigurointiin saatiin Ohjelman 2 mukaisella *mbed_app.json*

-tiedostolla. Makrolla *CELLULAR_DEVICE=QUECTEL_BG96* saadaan Mbed-OS käyttämään halutun modeemin ajuria [6]. Avnet-laajennusalustan käyttämät sarjayhteyskytkentänastat on kytketty STM-kehityslaudan rimaliittimen kytkentänastoihin *D8* ja *D2* [5], joten ne voitiin määrittää makroilla *MDMTXD=D8* ja *MDMTXD=D8*. Yleisen raudan abstraktointirajapinnan sisällyttämiseksi konfiguraatiotiedostoon piti lisätä ominaisuus *COMMON_PAL*, johon piti määritellä modeemin baudinopeus arvoon 115200. Oikea baudinopeus löytyi kokeilemalla yrityksen ja erehdyksen kautta.

```
void create_interface(const char *server_address,
                      void *handler=NULL)
{
    _server_address = server_address;
    _interface = M2MInterfaceFactory::create_interface(
        *this,
        _endpoint_name,              // endpoint nimi
        "test",                      // endpoint tyyppi
        lifetime,                    // elinika
        _port,                       // verkkoportti
        MBED_DOMAIN,                 // domain, verkkosivu
        SOCKET_MODE,                 // UDP/TCP
        NETWORK_STACK,               // verkkoprotokollat
        "");                         // context address string
    _interface->set_platform_network_handler(handler);
}

M2MSecurity* create_register_object()
{
    M2MSecurity *security;
    *security = M2MInterfaceFactory::create_security(
        M2MSecurity::M2MServer);
    security->set_resource_value(
        M2MSecurity::M2MServerUri, _server_address);
    security->set_resource_value(
        M2MSecurity::SecurityMode, M2MSecurity::Psk);
    security->set_resource_value(
        M2MSecurity::ServerPublicKey, _psk_id,
        _psk_id_length);
    security->set_resource_value(
        M2MSecurity::PublicKey, _psk_id, _psk_id_length);
    security->set_resource_value(
        M2MSecurity::Secretkey, _psk, _psk_length);
    return security;
}
```

Ohjelma 3. *Mbed_client-luokan metodit LwM2M-rajapinnan ja rekisteröintiobjektin luomiseen*

Laitteen on tarkoitus siirtää dataa LwM2M-protokollan mukaisesti toimimalla protokollan asiakas-osapuolena (engl. *Client*). Mbed-OS:n versio 5 tarjosi LwM2M-Client-kirjastototeutuksen nimeltä Mbed-client [19], jonka funktioilla kyseinen asiakasohjelma toteutettiin. Ohjelmassa 1 esitetään funktiot ja niiden kutsujärjestys, millä saadaan LwM2M-Client alustettua. Ohjelmassa 1 metodilla *create_interface* asetetaan Mbed-Clientille sen tarvitsema verkon käsittelijä, joka luotiin *easy_connect()*-funktiolla, ja palvelimen osoite, jotta laite kykenee rekisteröitymään oikealle LwM2M-palvelimelle. Seuraavaksi luodaan LwM2M-resurssimallin mukaiset objektit, jotka halutaan sisällyttää laitteeseen metodeilla *create_register_object* ja *create_device_object*. Resurssimalli toteutetaan luomalla kaikista yksittäisistä objekteista rekisteröintiobjektia lukuun ottamatta *M2MObjectList*-olio. Ohjelmasta on jätetty muiden objektien luominen esittämättä yksinkertaisuuden vuoksi. Lopuksi LwM2M-Client rekisteröidään palvelimelle objektilistoineen, jonka jälkeen se vastaanottaa asynkronisesti LwM2M-palvelimelta tulevia pyyntöjä omassa suoritussäikeessään.

Ohjelmassa 3 esitetään tarkemmin edellisessä kappaleessa mainitut metodit LwM2M-rajapintaolion ja rekisteröitiin vaadittavan turvallisuusolion luomiseen. Rajapintaolion luomismetodilla saadaan asetettua, minkä nimisenä ja tyyppisenä laite näkyy LwM2M-palvelimelle rekisteröityessään, sekä kuinka usein laite lähettää uudelleenrekisteröitymisviestin. Lisäksi sillä asetetaan verkkoon liittyvä konfiguraatio, johon sisältyy verkkoportti, käytetäänkö TCP- vai UDP-yhteyttä, mitä Mbedin protokollakirjastoa käytetään ja mihin verkko-osoitteeseen laitteen on tarkoitus kirjautua. Turvallisuusolion tarkoitus on sisältää laitteen verkkoliikenteen salaukseen liittyvät tiedot. Työssä valittiin salausmetodiksi etukäteen jaettava salausavain, jolloin salaus-ID ja salausavain ovat kovakoodattuna laitteeseen, ja vastaava ID–avain-pari on palvelimen tiedossa ennen yhteyden muodostamista.

Laitteelta palvelimelle lähetettävä data oli tarkoitus saada BLE-yhteyden kautta muilta laitteilta. Mbed-OS tarjoaa BLE-kirjastototeutuksen [3], joka mahdollistaa laitteen toimimisen BLE-protokollan keskus- tai oheislaitteena. Käytettävälle BLE-laajennusalustalle on myös valmis ajuri, joka saadaan käyttöön Ohjelman 2 *mbed_app.json*-tiedoston “NUCLEO_L476RG”-lohkolla. BLE-kirjastototeutuksen käyttäminen on tehty helpoksi ja suoraviivaiseksi, kuten on esitetty Ohjelmassa 4. BLE-olio rakennetaan kirjaston *Instance()*-funktiolla, jonka jälkeen olion metodilla *init()* voidaan alustaa Bluetooth-piiri valmiiksi kommunikoimaan muiden laitteiden kanssa. Alustusfunktiolle annetaan parametriksi Callback-funktio, jota kutsutaan alustuksen valmistuttua. Kyseisen funktion on lähinnä tarkoitus tarkistaa alustuksen onnistuminen ja asettaa tulevien skannausten parametrit. Varsinainen lähialueilla lähetettävien BLE-mainosviestien skannaaminen aloitetaan *startScan()*-metodilla, joksi parametrina annettavaa Callback-funktiota kutsutaan, jokaisen mainosviestin vastaanottamisen yhteydessä. Kyseinen *addvetisementCallback()*-funktio saa parametrina osoittimen mainosviestin sisältöön, joten siinä voidaan käsitellä data halutuilla toimenpiteillä.

Työssä skannaamista ei haluttu pariston säästämiseksi suorittaa jatkuvasti, vaan ohjelmaan sisällytettiin palvelimelta konfiguroitavissa oleva muuttuja *scantime*, joka määrää, kauanko skannaus on päällä ennen kuin BLE sammutetaan metodilla *shutdown()*.

```
void advertisementCallback(
    const Gap::AdvertisementCallbackParams_t *params)
{
...
}

void bleInitComplete(
    BLE::InitializationCompleteCallbackContext *params)
{
...
}

int main(){
    ...
    BLE &ble = BLE::Instance();
    ...
    while(true){
        ble.init(bleInitComplete);
        ble.gap().startScan(advertisementCallback);
        t.start();
        while(t.read_ms() < scantime ){
            ble.waitForEvent();
        }
        ble.shutdown();
        ...
    }
}
```

Ohjelma 4. *mbed BLE luokan käyttö Bluetooth keskuslaitteessa*

Työssä päädyttiin toteuttamaan datan lähetys palvelimelle ensirekisteröinnin jälkeen siten, että tietyin väliajoin laite lähettää niin sanotun uudelleenrekisteröintiviestin palvelimelle kertoakseen olevansa edelleen hengissä. Kyseinen viesti on vaadittu LwM2M-protokollassa, ja ilman sen lähettämistä palvelin unohtaa rekisteröityneen laitteen erikseen määrätyn aikakatkaisun (engl. *timeout*) perusteella. Rekisteröinnin päivitysviestin lähettämisen jälkeen laitteen on tarkoitus suorittaa BLE-skannaus ja lähettää mielenkiintoisten BLE-majakoiden tiedot eteenpäin palvelimelle. Rekisteröintitiheyteen ja siten myös datan lähetystiheyteen sekä BLE-skannausaikaan haluttiin voida vaikuttaa etänä LwM2M-palvelimen kautta. Tätä varten Mbed-Clientiin lisättiin oma objekti, joka sisälsi jokaista parametria kohden yhden resurssin kyseisen parametrin muuttamiseksi. Samaan objektiin sisällytettiin myös resurssi mielenkiintoisista MAC-osoitteista, joten LwM2M-palvelin kykenee myös valitsemaan,

mistä MAC-osoitteista lähetetyt BLE-mainosviestit tulisi lähettää Nb-IoT-verkossa palvelimelle asti ja mitkä jättää lähettämättä. Myöhemmin objektiin sisällytettiin vielä tieto laitteen virtalähteen jännitteestä. Kyseisen objektin luominen on esitetty Ohjelmasssa 5.

```
ble_object = M2MInterfaceFactory::create_object("26300");
M2MObjectInstance* ble_inst =
    ble_object->create_object_instance();

interval_res_ = ble_inst->create_dynamic_resource(
    "1", "IdleTime[min]", M2MResourceInstance::INTEGER,
    false);
interval_res_->set_operation(M2MBase::GET_PUT_ALLOWED);
interval_res_->set_value((uint8_t*)"1", 1);

window_res_ = ble_inst->create_dynamic_resource(
    "2", "ScanTime[ms]", M2MResourceInstance::INTEGER,
    false);
window_res_->set_operation(M2MBase::GET_PUT_ALLOWED);
window_res_->set_value((uint8_t*)"900", 3);

mac_res_ = ble_inst->create_dynamic_resource(
    "3","macsToScan", M2MResourceInstance::STRING,
    false);
mac_res_->set_operation(M2MBase::GET_PUT_ALLOWED);
mac_res_->set_value((uint8_t*)"0", 1);

battery_res_ = ble_inst->create_dynamic_resource(
    "4","BatteryVoltage", M2MResourceInstance::INTEGER,
    true);
battery_res_->set_operation(M2MBase::GET_ALLOWED);
battery_res_->set_value((uint8_t*)"0", 1);
```

Ohjelma 5. *lwM2M-resurssien luonti laitteen konfiguroimiseksi*

Koska Mbed-Client toimii asynkronisesti omasssa suoritussäikeessään, tapahtuu edellisen kappaleen mainitsemien parametrien päivittyminen laitteen pääohjelman While-silmukasta riippumatta, kun palvelimelta tehdään päivityspyyntö niihin. Tämän seurauksena laitteen pääohjelmassa jouduttiin lukemaan ennen BLE-skannausta skannauksen kestoon liittyvä parametri vastaavasta LwM2M-resurssista, koska se saattoi olla muuttunut edellisen skannauksen jälkeen, ja skannaamaan parametrin keston mukainen aika. Vastaavasti ennen laitteen uudelleenrekisteröintiviestiä laite odottaa tietyn idle-ajan, jonka pituus luetaan sitä vastaavasta LwM2M-resurssista.

Verkkoliikenteen vähentämiseksi haluttiin rajata, minkä BLE-laitteiden mainosviestit lähetetään edelleen LwM2M-palvelimelle ja mitkä jätetään huomiotta. Tätä varten mielenkiintoisia MAC-osoitteita varten oli oma LwM2M-resurssi BLE-objektissa.

Viestien suodattaminen ei kuitenkaan ollut ihan yhtä suoraviivaista kuin skannaus tai idle-ajan lukeminen resurssista, sillä palvelin tarvitsee tiedon kaikista BLE-laitteista, joita ylipäätään on NB-IoT-laitteen lähistöllä. Tämä ongelma päädyttiin ratkaisemaan siten, että Gateway-laite pitää kirjaa listarakenteen muodossa kaikista sen ympäristössä olevista MAC-osoitteista. Aina sen havaitessa uudesta MAC-osoitteesta tulevan BLE-mainosviestin lähettää se viestin automaattisesti palvelimelle. Tällöin palvelin saa yhden viestin kaikkia Gatevayn-havaitsemia laitteita kohden ja tietää niiden olemassaolosta. Tunnetun MAC-osoitteen tapauksessa laite lähettää viestin eteenpäin vain, jos se on kirjoitettuna BLE-objektin LwM2M-resurssiin palvelimen kirjoittamana. Lisäksi Gateway pitää kirjaa myös MAC-osoitteita jokaista skannauskertaa kohden. Tällä varmistetaan, että yhden laitteen mainosviesti käsitellään vain kerran yhden skannauskerran aikana.

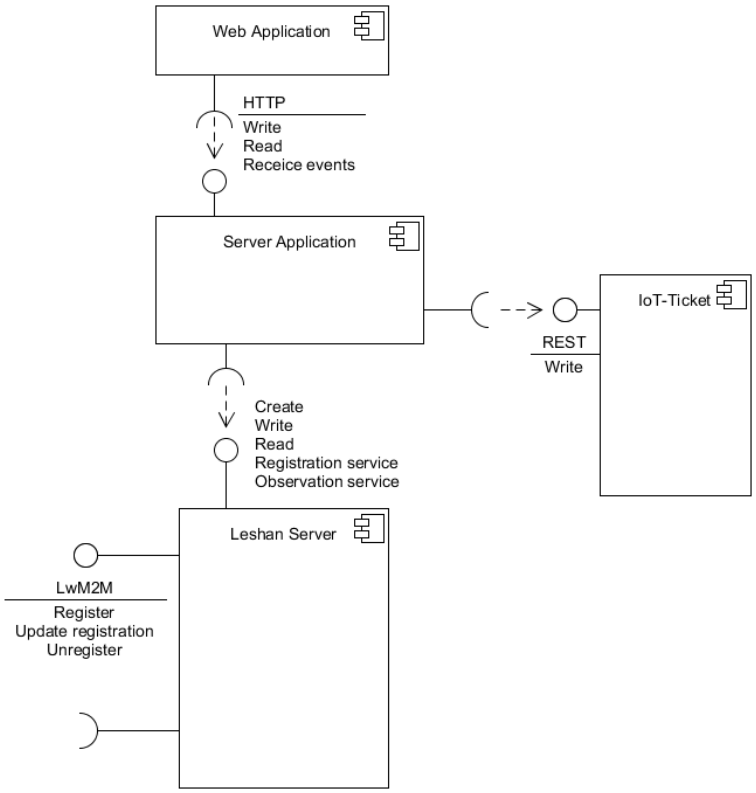
Quectelin NB-IoT-modeemin käyttöön liittyi yksi keskeinen ongelma: Tietyn verkossaoloajan jälkeen modeemi sekosi siten, että se lakkasi vastaanottamasta viestejä palvelimelta. Viestien lähetys näytti onnistuvan edelleen, mutta laite epäonnistui vastaanottamaan sille palvelimelta lähetetyt varmistusviestit toimittuaan hetken aikaa oikein. Epäselväksi jäi, johtuiko onglema ohjelmointivirheestä ajurissa, joka vastasi prosessorin ja modeemin välisestä AT-kommunikaatiosta, vai oliko ongelma itse modeemin Firmwaressa. MBed-Client koitti ratkoa ongelmaa tekemällä lähetyksen uudelleenyrityksiä, mutta ei kyennyt selviämään tästä ongelmasta, joka johti laitteen kaatumiseen. Ongelman juurisyytä ei ratkaistu, mutta ongelema kierrettiin toteuttamalla uudelleenrekisteröinti-funktioon aikakatkaisu. Laite odotti rekisteröitymisen onnistumisesta aiheutuvaa ilmoitusta (engl. *event*) tietyn ajan, ja laitteen epäonnistuessa vastaanottamaan rekisteröintivarmistuksen kyseisessä ajassa suoritti laite Resetin. Jotta palvelin havaitsee laitteessa tapahtuneen Resetin, laitteeseen lisättiin LwM2M-resurssi indikoimaan Resetiä, johon kirjoitettiin 1 aina Resetin tapahtuessa, ja palvelimen tehtäväksi jäi resurssin nollaaminen. Reset resurssin nollaamisen yhteydessä LwM2M-palvelimen tehtäväksi jäi myös kirjoittaa kaikki konfiguraatioresurssit uudelleen, sillä kyseinen tieto katoaa laitteelta resetin yhteydessä.

Toinen MBed Clientin käyttöön liittyvä ongelma on, että se ei ilmoita sovelluskerrokseen dataviestien lähetyksen onnistumisesta. Lisäksi Mbed Client puskuroi automaatisesti CoAP-viestejä sitä mukaa, kun sovellus tekee LwM2M-resursseihin muutoksia, lähettää viestit välittömästi ja pitää niitä puskurissa uudelleenlähetystä varten, kunnes saa viesteihin vastauksen. Tämä menetelmä toimii varmasti moitteettomasti esimerkiksi ethernet-yhteyttä käytettäessä, jolloin viestejä tulee nopeasti ja pakettien katoamista tapahtuu harvoin. NB-IoT:n tapauksessa viestien lähetys–vastaanotto-sykli kestää kuitenkin kauan, joten riskinä on täyttää lähetyspuskuri sillä, että muuttaa sovelluskerroksessa LwM2M-resurssien arvoja liian tiheään tahtiin. Datan lähetys tapahtui tämän työn tapauksessa BLE-mainos LwM2M-resurssiin kirjoittamalla. Viestiliikenteen hitaudesta aiheutuvaa ongelmaa pyrittiin minimoimaan siten, että BLE-skannauksen aikana ensin puskuroitiin 10 laitteen mainosviestit, ja

skannauksen jälkeen nämä viestit kirjoitettiin mainosresurssiin yhden sekunnin viiveillä toisistaan. Viestiliikenteen hitauden takia päädyttiin myös ratkaisuun, jossa LwM2M lähettää kaikki mahdolliset konfigurointiviestit heti onnistuneen rekisteröitymisen jälkeen, joten laitteen pääohjelmaan sisällytettiin viiden sekunnin viive rekisteröitymisen ja dataviestien väliin, jotta vältyttiin samanaikaisten viestien lähettämiseltä.

3.2.2 Palvelimen ohjelmisto

Palvelimen osuus kokonaisarkkitehtuurista on esitetty Kuvassa 27. Palvelimen tarkoituksena on toimia LwM2M-protokollan palvelinosapuolena ja vastaanottaa rekisteröintejä LwM2M-asiakkailta. Tässä työssä asiakkaina on vain yksi edellä kuvattu BLE-gateway, mutta palvelin on toteutettu niin, että se pystyy ottamaan vastaan rekisteröintejä myös useammalta asiakkaalta. Rekisteröinnin yhteydessä palvelin lähettää automaattisesti Observe-pyynnön, jonka seurauksena se alkaa saada rekisteröityneeltä laitteelta BLE-mainosdataa. Palvelin ei säilö vastaanottamaansa dataa, vaan lähettää sen edelleen IoT-Ticketiin, mikäli katsoo sen oleelliseksi.



Kuva 27. *Palvelimen arkkitehtuuri*

Palvelin tarjoaa datan edelleenlähettämisen lisäksi selainrajapinnan, josta rekisteröityneen laitteen parametreja voi muuttaa. Selaimen kautta voi esimerkiksi määrittää, mikä on haluttu BLE-gatewayn skannausaika, jonka se käyttää mainosviestien skannaamiseen, tai mikä on idle-aika, jonka laite viettää kommunikoimatta palvelimen

kanssa energian säästämiseksi. Lisäksi käyttöliittymästä voidaan määrittää, mitkä BLE-majakoiden MAC-osoitteet ovat kiinnostavia eli mistä Gatewayn ympäröimistä BLE-laitteista peräisin olevat mainosviestit on järkevää lähettää edelleen palvelimelle ja IoT-Ticketiin palvelimen toimesta. Palvelimen on tarkoitus myös parsia BLE-mainosviesteistä olennainen informaatio ihmisen luettavaan muotoon ennen IoT-Ticketiin lähetystä.

Palvelin on toteutettu pitkälti avoimen lähdekoodin Eclipse-Leshan kirjastoja käyttäen. Leshan tarjoaa LwM2M-palvelimen esimerkkisovelluksen [14], jossa Back-end on toteutettu Java-ohjelmointikielellä ja Front end Java scriptillä. Palvelin on toteutettu pitkälti kyseisen esimerkkisovelluksen pohjalta muokaten siten, että se soveltuu paremmin tämän työn ongelmaan. Kuten Kuvasta 27 nähdään, käyttää palvelin LwM2M-protokollaan Leshan Serverin kirjastoja ja selainrajapintaa. Lisäksi sovellus käyttää IoT-Ticketin REST-rajapintaa datan edelleenlähettämiseen.

Leshanin esimerkkiohjelma, johon työssä toteutettu palvelinohjelma pohjautuu vahvasti, toimii geneerisenä LwM2M-palvelimena erityyppisille LwM2M-asiakkaille. Asiakkaat voivat rekisteröityä siihen, mikäli niillä on tiedossa palvelimen osoite, ja rekisteröitymisviestin jälkeen palvelin pitää kirjaa asiakkaasta ja sen rekisteröinnin yhteydessä listaamista resursseista. Palvelin ei kuitenkaan pidä kirjaa resurssien sisältämästä datasta. Rekisteröinti säilyy palvelimen muistissa rekisteröintiviestiin sisällytetyn ajanjakson ajan, mikäli asiakas ei lähetä uudelleenrekisteröintiviestiä ennen rekisteröintiajan umpeutumista. Mikäli asiakaslaite epäonnistuu lähettämään uudelleenrekisteröintiviestin, joutuu se suorittamaan täyden rekisteröinnin uudelleen sisältäen mahdollisen salauksen uusimisen.

Esimerkkiohjelmassa on valmis selainkäyttöliittymä, jossa on listattuna kaikki palvelimeen kulloinkin rekisteröityneenä olevat asiakaslaitteet. Jokainen listan asiakas on hyperlinkki sivulle, jossa on listattuna kaikki kyseisen laitteen resurssit LwM2M-resurssimallin mukaisesti. Käyttöliittymä tarjoaa painikkeet resurssin lukemiseen, kirjoittamiseen, seuraamiseen ja seuraamisen lopettamiseen. Painikkeiden painaminen saa palvelimen lähettämään asiakasohjelmalle LwM2M-lukukyselyn, ja mikäli asiakas vastaa, tulee vastaus näkyviin käyttöliittymään. Tarkkailuviestin lähetys aloittaa jatkuvan datankeruun, jossa asiakkaalta tulee uusi vastaus aina, kun tarkkailtavassa resurssissa tapahtuu muutos.

Muutokset Leshan esimerkkipalvelimeen

Esimerkkipalvelinta ei voitu käyttää sellaisenaan, vaan sitä piti muokata tämän työn vaatimusten mukaiseksi. Esimerkkiohjelmassa laitteen rekisteröinti tapahtuu automaattisesti, mutta käyttäjän täytyy aina aloittaa datan keräys selainrajapinnan kautta. Tämä osoittauitui erityisen ongelmalliseksi, sillä laitteen modeemin ongelmien takia laite ei pysynyt jatkuvasti rekisteröityneenä. Tämän sijaan laitteen oli suoritettava jokaisen resetin jälkeen uusi rekisteröityminen. Esimerkkiohjelma olisi tässä tilanteessa

edellyttänyt, että jokaisen uudelleenrekisteröinnin jälkeen käyttäjä olisi selaimen kautta aloittanut mielekkään resurssin tarkkailun uudelleen. Lisäksi laitteen resetoituessa sen konfigurointiin liittyvät resurssit nollautuvat, joten käyttäjän olisi pitänyt myös kirjoittaa konfiguraatioresurssien arvot uudestaan.

Ongelma päädyttiin ratkaisemaan siten, että laitteen ulosrekisteröinnin yhteydessä palvelin ei unohda rekisteröintiä kokonaan, vaan tallentaa ulosrekisteröityneen asiakkaan omaan tietorakenteeseensa. Lisäksi laitteen konfiguraatiota varten luotiin oma tietorakenne, jossa on muistissa viimeisimmät asiakkaalle lähetetyt viestit. Palvelin voi palauttaa laitteessa olevat viimeisimmät asetukset näiden tietorakenteiden perusteella automaattisesti, mikäli laite on joutunut resetoitumaan. Lisäksi käyttäjä voi verkkorajapinnan kautta asettaa uusia asetuksia tai asettaa uusia resursseja datankeräyskohteiksi, vaikka laite ei olisi kyseisellä hetkellä rekisteröityneenä.

```
RegistrationListener reglistener = new RegistrationListener()
{
    public void registered()
    {
    ...
    }
    public void updated()
    {
    ...
    }
    public void unregistered()
    {
    ...
    }
}
ObservationListener obslistener = new ObservationListener()
{
    public void newObservation()
    {
    ...
    }
    public void onResponse()
    {
    ...
    }
}
lwServer.getRegistrationService().addListener(reglistener);
lwServer.getObservationService().addListener(obslistener);
```

Ohjelma 6. *Leshanin tapahtumakuuntelijat*

Leshan hoitaa LwM2M-viestittelyn asiakkaiden kanssa kapseloituna ja tarjoaa sovelluskerrokselle Ohjelmassa 6 tapahtumiin reagoimiseen *RegistrationListener*- ja

ObservationListener-luokat. Palvelinoliolle voidaan asettaa tällaisia kuuntelijaluokkia, joiden metodit voidaan ylikirjoittaa ja siten tehdä sovelluskohtaista toiminnallisuksia eri LwM2M-protokollan tapahtumien seurauksena. Kun rekisteröintikuuntelija on asetettu palvelinoliolle, tämä kutsuu sen metodia *registered()* automaattisesti, kun jokin LwM2M-asiakas suorittaa onnistuneen rekisteröinnin. Vastaavasti asiakkaan lähettäessä dataa palvelimelle tarkkailupyynnön seurauksena palvelin kutsuu tarkkailukuuntelijan metodia *onResponse()*.

Jotta työssä saatiin huomioitua laitteen ogelmat rekisteröinnin ylläpidossa ja toisaalta kyettiin muuttamaan laitteen konfiguraatiota sen nukkuessa, luotiin Leshanin esimerkkiohjelmaan uusi rekisteröinnin kuuntelija. Uuden rekisteröinnin tapahtuessa *registered()*-metodissa tarkastetaan ensin, onko kyseinen laite rekisteröitynyt jo aijemmin. Laitteen ollessa ensikertalainen luodaan sitä varten tarvittavat tietorakenteet konfiguroinnin uusimiseksi. Seuraavaksi lähetetään laitteen BLE-resurssiin tarkkailupyyntö, jotta saadaan laitteelta dataa sen keräämistä BLE-mainoksista. Lopuksi lähetetään laitteelle uudet konfiguraatioparametrit, mikäli käyttäjä on muuttanut niitä laitteen ollessa unessa tai mikäli laite on joutunut suorittamaan resetin ja sen konfiguraatio on alustusarvossa. Rekisteröinnin päivittyessä laite ei ole voinut resetoitua, joten sille lähetetään uudet asetukset vain, jos käyttäjä on vaihtanut niitä laitteen nukkuessa. Rekisteröinnin loppuessa laitteen tiedot tallennetaan tietorakenteeseen uudelleenrekisteröinnin varalta.

Toinen merkittävä muutos esimerkkiohjelmaan liittyi datan keräykseen. Esimerkkiohjelman esittäessä laitteen resurssien arvoja ainoastaan selainkäyttöliittymässä haluttiin työn ohjelmassa edelleenlähettää valikoidut arvot IoT-Ticketiin. Edelleenlähettämistä varten palvelinolioon lisättiin *ObservationListener*-olio ja tietorakenne, jonka perusteella ohjelma päättelee, tuleeko saapunut datapaketti edelleenlähettää vai ei. Palvelimen vastaanottama data on laitteen edelleenlähettämä BLE-mainosviesti. Viesti sisältää alkuperäisen BLE-oheislaitteen MAC-osoitteen, joka toimii myös edelleenlähettämisperusteena palvelimen tietorakenteessa. Tarkkailukuuntelijan *onResponse()*-metodissa parsitaan ensin vastaanotettu BLE-mainosviesti ja lisätään siihen aikaleima. Tämän jälkeen tarkastetaan, onko kyseisestä MAC-osoitteesta vastaanotettu data asetettu edelleenlähetettäväksi. Mikäli data halutaan edelleenlähettää, ohjelma rakentaa siitä IoT-Ticketin REST-rajapinnan funktioilla viestin ja lähettää sen eteenpäin.

Bluetoothin parsimista varten ohjelmaan kirjoitettiin myös oma luokka. Luokan oli tarkoitus muodostaa eri laitteiden mainosviesteistä eriteltyä, niiden sisältämää dataa ihmisen luettavaan muotoon. Eri laitteet käyttävät BLE-viestiensä sisältöihin erilaisia protokollia, joten tämän työn palvelimeen toteutettiin kolme formaattia, jotka se osasi parsia perustuen käytettäviin testimajakoihin. Pääasiallinen testimajakka oli Ruuvitag, jonka viestien data voi olla kahdessa eri formaatissa. Toinen on Googlen kehittämä yleisformatti nimeltä Eddystone, jonka URI-kehykseen on koodattu majakan lämpötila, ilmanpaine ja ilmankosteuslukemat. Toinen formaateista on Ruuvitagin oma raakadata,

jossa mainosviestiin sisältyy edellisten lisäksi majakan kiihtyvyysanturien lukemat kolmen eri akselin suhteen. Esimerkkimajakkoina käytettiin myös Cypress Energyn Cyalkit -majakoita, joiden sisältämät ilman lämpötila ja -kosteuslukemat on koodattu Applen kehittämän Ibeacon-viestiformaatin loppuosaan. Muiden BLE-majakoiden viestit parsija kykenee erottelemaan Ibeaconeiksi tai Eddystone-viesteiksi eli parsimaan osittain tai muiden formaattien tapauksessa ei ollenkaan. Osittain tai ei ollenkaan parsittavat viestit ohjelma tallentaan vain raakadatana.

Jotta käyttäjä kykenee valitsemaan, mistä MAC-osoitteista lähtöisin olevat mainosviestit tulee kerätä IoT-Ticketiin, esittää ohjelma kerätyt ja parsitut kustakin osoitteesta viimeisimpänä vastaanotetut mainosviestit selainkäyttöliittymässä. Esimerkkiohjelmassa palvelin kommunikoi selainkäyttöliittymän kanssa Java-Servlet-tiedostojen avulla. Asiakkaan tietojen esittämiseen tarvittavien HTTP-kyselyiden käsittelyyn tehtyyn *ClientServlet*-tiedostoon piti lisätä käsittelijä kaikkien palvelimen tietämiä MAC-osoitteita vastaavat viimeisimmät BLE-mainokset. Lisäksi piti lisätä käsittelijä, jolla voitiin lisätä tai poistaa kukin MAC-osoite edelleen lähetettävien mainosten joukkoon. Näiden avulla voidaan selainkäyttöliittymistä selata, millaisia BLE-majakoita palvelimelle rekisteröityneen Gateway-laitteen läheisyydessä on, ja valita niiden joukosta mieleiset, joiden data halutaan kerätä IoT-Ticketiin.

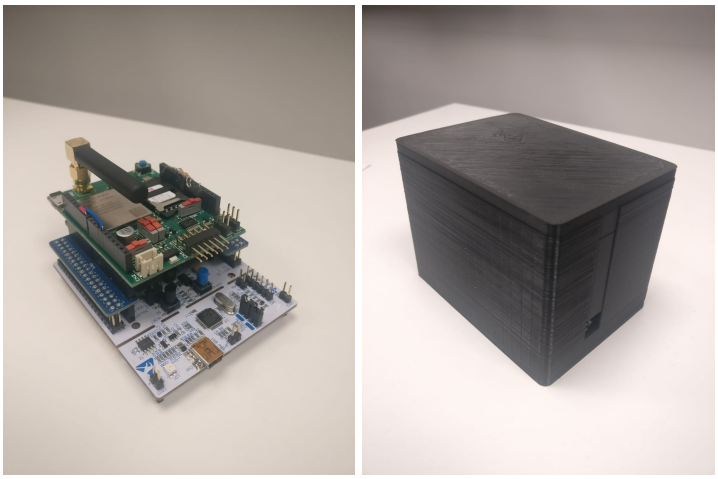
Iot-Ticketin tarjoama REST-rajapinta viestien edelleenlähetykseen vaatii kirjautumistunnukset jollekin tilille. Kirjaustunnuksella varmistetaan, että rajapinnan käyttäjällä on oikeus lähettää dataa, ja siten suojaudutaan palvelunestohyökkäykseltä. Toisaalta niillä määritetään, minkä käyttäjän tililtä kerättävää dataa on mahdollisuus tarkastella. Jotta Käyttäjätietoja ei tarvinnut kovakoodata tietyllä käyttäjätunnuksella, lisättiin selainkäyttöliittymään uusi välilehti myös sisäänkirjautumista varten. Sisäänkirjautumisen jälkeen palvelin kykenee käyttämään REST-rajapintaa ja käyttäjä kykenee aloittamaan datan edelleenlähetyksen.

4. TULOKSET

Tässä luvussa käydään läpi lopputulos eli se, mitä saatiin aikaiseksi. Elektroniikan osalta aikaansaannos on edellisessä luvussa kuvattujen piirilevyjen kytkeminen ja kotelointi. Ojelmiston osalta tulos on kokonainen dataputki BLE-majakoilta LwM2M-asiakasojelman ja -palvelimen kautta Wapicen IoT-Ticket-pilveen. Luvussa käsitellään myös ilmenneitä ongelmia ja pohditaan kehitysehdotuksia siitä, mitä olisi voitu tehdä toisin.

4.1 Valmis järjestelmä

Kuvassa 28 (a) on esitetty lopullinen protoyyppi BLE-NB-IoT-yhteyspisteenä toimivasta laitteesta. Laitteessa alimpana piirilevynä on mikrokontrollerin sisältävä STM Nucleo L476RG -kehityslaitealusta. Keskimmäisenä on Bluetooth-modeemin sisältävä STM X-NUCLEO-IDB05A1 -laajennusalusta. Päällimmäisenä kytkennässä on NB-IoT-modeemin sisältävä *Avnet Silica NB-IoT Sensor Shield* -laajennusalusta, jonka kytkentänastoista Luvussa 3 mainitut on katkaistu. Kuvasta nähdään myös hyppylangat vasemmanpuoleisessa kytkentärimassa, jolla katkaistut NB-IoT-laajennusalustan tarvitsemat nastat on uudelleenreititetty. Oikeanpuoleisessa kytkentärimassa on tehty jännitejako 2 vastuksen kautta 5 V:n ja maan välille, jotta virtalähteen jännitettä voidaan mitata. Lopuksi laitteen kotelolle tehtiin 3D-malli, joka tulostettiin 3D-tulostimella. Kuvassa 28 (b) laite on koteloituna tulostetussa kotelossa. Kotelo on umpinainen lukuun ottamatta reikää, jonka kautta voidaan kytkeä USB-kaapeli.



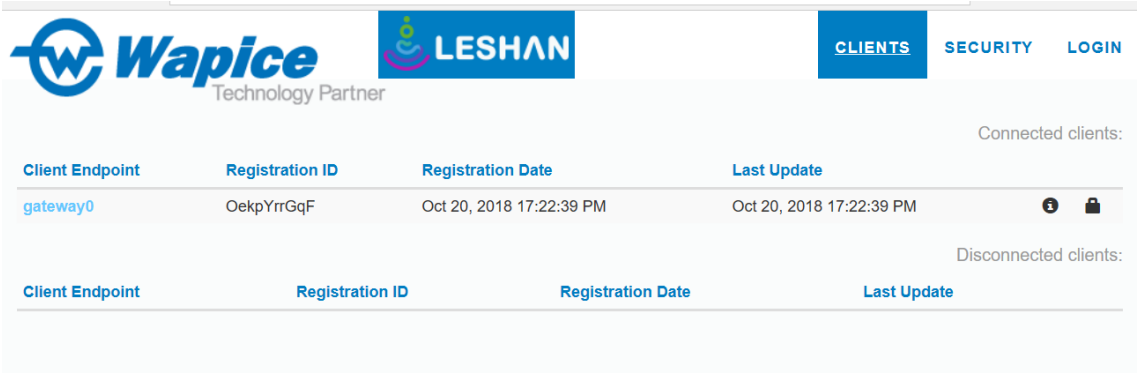
(a) *Valmis BLE-NB-IoT-yhteyspiste*

(b) *Laite koteloituna*

Kuva 28. *Kytketty ja koteloitu prototyyppi*

Laitteen ohjelma muodostaa internetyhteyden NB-IoT-verkolla, jonka jälkeen se kirjaa itsensä LwM2M-asiakkaana palvelimelle. Palvelimeen kirjauduttuaan laite suorittaa säännöllisesti BLE-skannauksen ja lähettää vastaanottamansa mainosviestit eteenpäin palvelimelle MAC-osoitteista, jotka ovat sille uusia. Tämä tehdään, jotta palvelimelta nähdään, millaisia BLE-laitteita yhteyspisteen lähistöllä on ja mitkä ovat kyseisten laitteiden MAC-osoitteet. Palvelimelta voi konfiguroida laitteen skannaukseen käyttämän ajan ja periodin, kuinka usein skannaus suoritetaan. Lisäksi palvelimelta voi määrittää MAC-osoitteet, joista asiakasohjelma lähettää myös ensimmäisen vastaanotetun mainosviestin jälkeen vastaanotetut mainosviestit eteenpäin, jotta palvelin voi kerätä niiden datan talteen. Laite myös selviää usein ilmenevistä NB-IoT-modeemin sekoamisista suorittamalla Resetin, mikäli rekisteröinnin uusimisviestin timeout ylittyy.

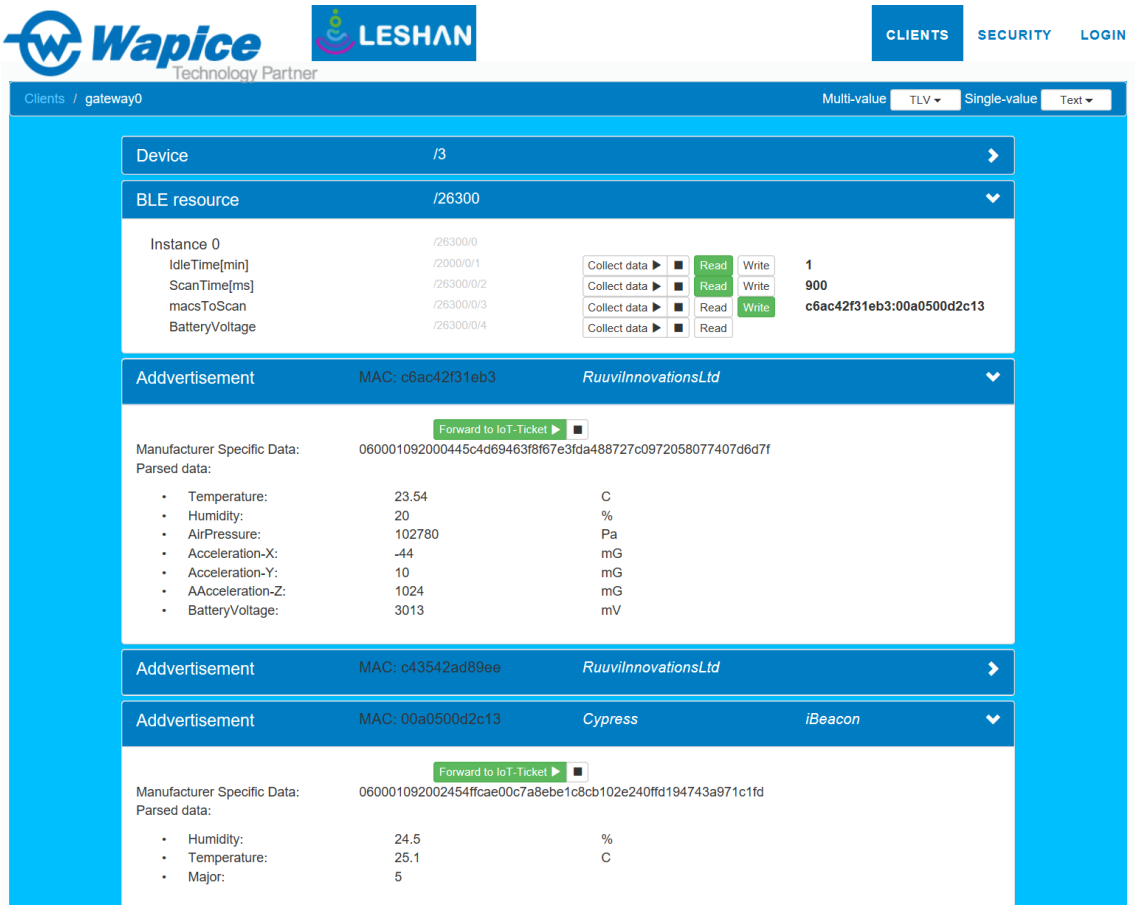
Kuvassa 29 on esitetty LwM2M-palvelimen verkkokäyttöliittymän välilehti, jossa on listattuna kaikki palvelimelle rekisteröityneet LwM2M-asiakaslaitteet. Kuvassa palvelimelle on rekisteröitynä ainoastaan ”gateway0”-niminen edellä esitetty prototyyppi, joka on listattuna parhaillaan kirjautuneina olevien laitteiden listassa (engl. *Connected clients*). Mikäli asiakasohjelma epäonnistuu uusimaan rekisteröintinsä tietyn rekisteröintiperiodin sisällä LwM2M-protokollan mukaan, palvelin unohtaa rekisteröityneen laitteen ja se häviää kirjautuneiden laitteiden listasta. Laitteessa havaittujen modeemiongelmien vuoksi kirjautumisen uusiminen epäonnistuu kuitenkin melko usein, joten palvelimeen tehtiin listaus myös kirjautumattomista laitteista (engl. *Disconnected clients*), jotta käyttäjä voi käsitellä laitetta myös sen epäonnistuessa uusimaan kirjautumisensa. Asiakaslaitteen nimi toimii listassa hyperlinkkinä uudelle verkkosivulle, jossa voidaan tarkastella kyseisen laitteen resursseja.



Kuva 29. *LwM2M-palvelimelle rekisteröityneet laitteet*

Kuvassa 30 on esitetty verkkokäyttöliittymän sivu, jossa on listattu kirjautuneen LwM2M-asiakkaan resurssit. Resursseissa on listattu suljettuna pakollinen ”Device”-LwM2M-objekti, jota ei tässä sovelluksessa käytetä mihinkään, ja ”BLE resource” -objekti avattuna. ”BLE resource” -objektin kautta voidaan lukea ja uudelleenkirjoittaa laitteen käyttämät parametrit, jotka määrittävät BLE-skannaukseen käytettävän ajan millisekunteina (*ScanTime[ms]*) ja laitteen idle-ajan minuutteina (*IdleTime[ms]*), joka määrittää laitteen skannausvälin. Lisäksi objektin kautta voidaan

kirjoittaa, mistä MAC-osoitteista palvelin haluaa vastaanottaa lisää BLE-dataa (*macsToScan*). BLE-objektin viimeinen resurssi *BatteryVoltage* kertoo BLE-yhteyspisteen virtalähteen jännitteen, joten sen voi ainoastaan lukea. Mikäli laite epäonnistuu vastaamaan luku- tai kirjoituspyyntöön, palvelin säilöö epäonnistuneen pyynnön, kunnes päätelaite uusii rekisteröintinsä, jonka jälkeen pyyntö lähetetään laitteelle uudelleen. Tämä mahdollistaa laitteen konfiguroinnin muutoksen, vaikka laite olisi pyynnön lähetyshetkellä unessa.

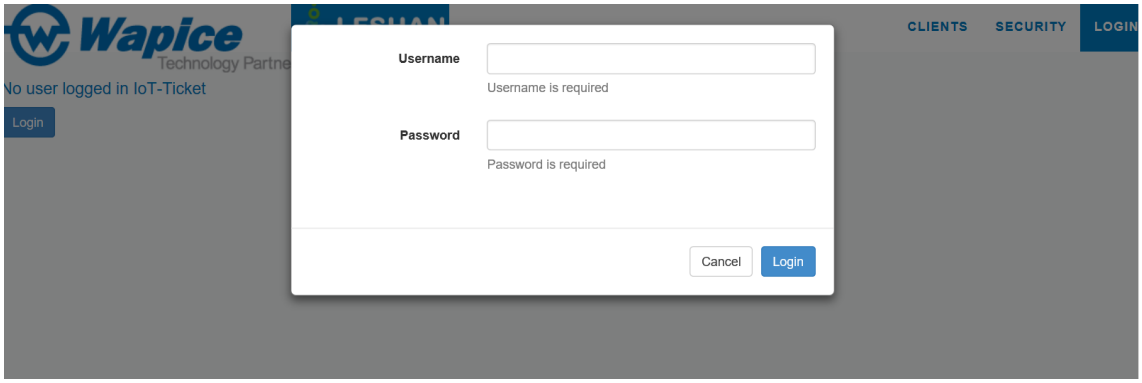


Kuva 30. *LwM2M-laitteen resurssit*

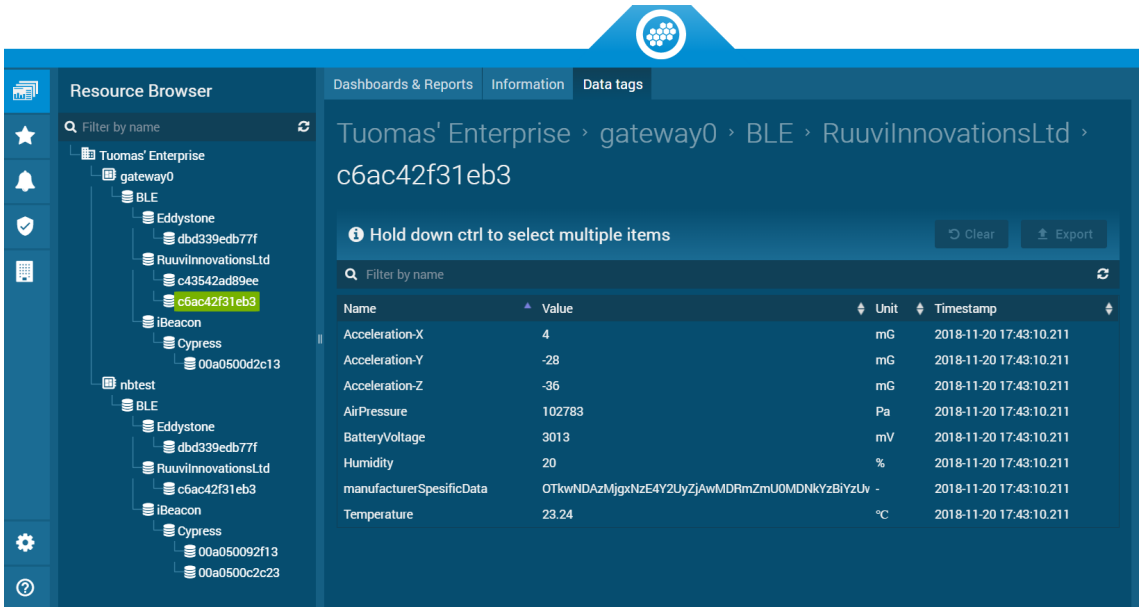
Laitteessa on ”BLE resource” -resurssin lisäksi myös mainosviestille oma objekti, jonka ainoana resurssina on yksittäinen BLE-mainosviesti, mutta se ei näy käyttöliittymässä sellaisenaan. Palvelin lähettää kyseiseen resurssiin rekisteröinnin yhteydessä automaattisen ”Subscribe”-pyynnön, jonka jälkeen laite lähettää palvelimelle resurssin arvon ”Notify”-viestinä aina arvon muuttuessa. Mainosobjektin näyttämisen sijaan UI näyttää kustakin yhteyspistettä ympäröineestä BLE-laitteesta viimeisimmän vastaanotetun mainosviestin. Käytännössä useimmat näistä ovat myös ainoat BLE-mainokset kyseisistä MAC-osoitteista, sillä yhteyspiste lähettää saman BLE-laitteen mainoksen eteenpäin vain kerran, ellei sitä ole erikseen kirjoitettu ”BLE resouce” -objectin *macsToScan*-resurssiin. Syy miksi yhteyspiste lähettää kaikista laitteista viestin kerran, on, että käyttöliittymästä nähtäisiin, mitä laitteita yhteyspisteen lähellä on, jotta käyttäjä osaa valita, mistä halutaan lisää dataa. Toisaalta turhan

datanlähetyksen välttämiseksi yhteyspiste lähettää mainosviestin kaikista MAC-osoitteista vain kerran.

Palvelin parsii mainosviestin osittain tai kokonaan, mikäli se tunnistaa käytetyn viestiformaatin. Käytännössä viestiformaatit, joista palvelin osaa parsia datan, ovat Ruuvitagien käyttämä raakaformaatti sekä Eddystone, josta erikoistapauksena palvelin osaa parsia datan ”https://ruu.vi”-alkuisista URI-viesteistä ja Cyppressin majakoissa käytetty versio iBeaconista. Käyttöliittymässä jokaisesta MAC-osoitteesta lähetetty data näytetään raakana *Manufacturer Specific Data* -rivillä sekä auki parsittuna, mikäli palvelin osaa sen parsia.



Kuva 31. *Käyttöliittymän kirjautumisikkuna*



Kuva 32. *Datanäkymä IoT-Ticketissä*

Avatussa mainosviestissä on myös mahdollista aloittaa datan edelleenlähetys halutun MAC-osoitteen BLE-mainoksista IoT-Ticketiin klikkaamalla painiketta ”Forward to IoT-Ticket”. Operaation onnistuminen vaatii, että Kuvan 31 mukaisella *SECURITY*-välilehdellä on ensin annettu palvelimelle jonkin IoT-Ticket-käyttäjän tunnukset, sillä IoT-Ticket API:n käyttämien viestien lähetys edellyttää autentikointia.

Onnistuneen tunnusten antamisen ja edelleenlähetyksen aloittamisen jälkeen LwM2M-palvelin edelleenlähettää kaikki kyseiseltä BLE-majakalta lähtöisin olevat viestit parsittuna IoT-Ticketiin. Lopullisessa määränpäässään IoT-Ticketissä data on ryhmitelty laitteittain ja protokollittain Kuvan 32 mukaisesti.

4.2 Havaitut ongelmat ja mitä opittiin

Ensimmäinen havaittu ongelma liittyi Quectelin modeemin sekoamiseen. Ohjelma sai onnistuneesti muodostettua verkkoyhteyden NB-IoT-verkossa, mutta satunnaisen ajanjakson jälkeen modeemi ei enää vastaanottanut saapuvia viestejä. Ongelma osoittautui haastavaksi ratkaista, sillä laite kykeni edelleen lähettämään viestejä. Viestien katoaminen on normaalia UDP-yhteyttä käytettäessä, joten Mbed Client asettaa viestit lähetysjonoon, jossa niitä uudelleenlähetetään tietyn kasvavan intervallin välein, kunnes viesteihin saadaan vastaus. Mbed Client ei siten suoranaisesti huomaa viestien vastaanoton epäonnistumista suoraan, mutta sekoaa, mikäli lähetysjono tulee täyteen. Clientiä muokattiin siten, että lähetysjonon täyttyessä se tyhjätään, mikä esti ohjelman sekoamisen, mutta ei korjannut vielä oikeaa ongelmaa.

Modeemin sekoamisongelman olisi voinut yrittää korjata sillä, että viestien lähetysjonon täyttyessä modeemin resetoisi ja alustaisi verkko-olion uudelleen. Mbed Client ei kuitenkaan tue suorituksen keskeyttämistä modeemin uudelleenalustamisen ajaksi vaan sekoaa, jos verkko-oliota yrittää vaihtaa kesken suorituksen. Tietysti koko Mbed Clientin olisi voinut yrittää alustaa uudelleen, mutta kyseisen olion tuhoaminen ei vapauta kaikkia sen käytössä olevia resursseja, joten jatkuva uudelleenalustaminen olisi johtanut muistin täyttymiseen. Myös kaikkien resurssien arvot olisi pitänyt tallettaa uudelleenalustamisen ajaksi.

Helpoin tapa ratkaista modeemin sekoaminen oli muuttaa ohjelman toimintaa siten, että ennen jokaista BLE-skannausta laite lähettää palvelimelle LwM2M-standardin mukaisen rekisteröinnin uusimisviestin aikakatkaisulla. Mikäli Client ei kykene vastaanottamaan kuittausta uudelleenrekisteröintiviestiin useista yrityksistä huolimatta tietyssä aikaikkunassa, resetoituu koko ohjelma ja alustuu uudelleen. Jotta palvelimelta kirjoitetut asetukset saadaan palautettua, asiakasohjelmassa on yksi resurssi, joka ilmoittaa tapahtuneesta resetistä. Palvelimen lukiessa kyseisen resurssin ja havaitessa resetin tapahtumisen se kirjoittaa uudelleen kaikki viimeisimmät asetukset BLE resource -objektiin.

Toinen suuri ongelma liittyy itse sovelluksen mielekkyyteen NB-IoT-teknologialla. Yleisesti LPWAN-tekniikoilla on tarkoitus tehdä yksikköhinnalta halpoja laitteita, jotka lähettävät hyvin yksinkertaista muutaman anturin dataa. Järkevästi tehtynä BLE-gateway-laitteen tulisi pariutua kaikkien kiinnostavien BLE-oheislaitteiden kanssa ja tehdä jokaiselle oma LwM2M Client, jossa olisi BLE-oheislaitteiden GATT-palveluita vastaavat LwM2M-objektit. Tällainen ratkaisu oli kuitenkin mahdoton käytetyllä

elektroniikalla, koska laitteen muisti ei olisi riittänyt usean Mbed Clientin ajamiseen samalla laitteella. Ongelmana parituksissa on myös kommunikaatioviiveet, mikäli halutaan, että NB-IoT-laite nukkuu suurimman osan ajasta. Kun palvelimelle kirjattaisiin lähetettäväksi parituskäsky, voisi mennä pitkään, ennen kuin se lähetetään, ja lähetyksen jälkeen käyttäjän pitäisi edelleen valita, mistä paritetun laitteen resursseista hän on kiinnostunut. Tästä syystä työssä päädyttiin vain skannaamaan mainosviestejä ja keräämään dataa laitteilta, jotka eivät vaadi paritusta.

Työtä lukiessa voi mietityttää, miksi siinä käytettiin niin paljon valmiskirjastoja sen sijaan, että esimerkiksi viestinvälitysprotokolla olisi ohjelmoitu kokonaan itse. Työssä ei käynyt suoraan ilmi se, että yksi mielenkiinnon kohde oli myös mahdollistaa muiden valmistajien Mbed-laitteiden liittäminen IoT-Ticketiin tulevaisuudessa, jolloin on luonnollista käyttää protokollaa, jota Mbed on itse kehittänyt käytettäväksi. Toisaalta Mbed-Client-kirjasto toteuttaa LwM2M-asiakasohjelman monimutkaisella tilakoneella, jonka monimutkaisuus johtuu pääosin viestien vastaanottamisen asynkroonisuudesta. Vastaavan työn tarkoitukseen räätälöidyn ohjelmakirjaston tekeminen olisi siis ollut vaativaa ja aikaa vievää, joten luultavasti työssä päästiin monessa kohtaa helpommalla käyttäen valmiskirjastoja ja sopeutumalla niiden rajoitteisiin.

Työ oli hyvin opettavainen, sillä tekijän tietoliikennetekniikan osaaminen oli työtä aloittaessa lähes olematon. Työtä tehdessä tuli kuitenkin opittua paljon siitä, miten viestit liikkuvat yleisesti internetissä ja millaisia haasteita rajoittuneet laitteet ja verkot tuovat siihen lisää. Jatko-opiskeltavaa työn jäljiltä jäi paljon ainakin salauksesta liittyen siihen, mitä salausalgoritmia ja menetelmää on missäkin tilanteessa järkevää käyttää ja onko salaaminen ylipäätään mahdollista, jos lähetettävän viestin pituus on muutama tavu. Työn myötä piti opetella myös uusi ohjelmointikieli *Java*, joka tosin muistutti paljon jo koulussa opittua ohjelmointikieltä *C++*:aa, sekä *JavaScript*-laajennus *Angular*, jotta verkkokäyttöliittymä saatiin muokattua halutunlaiseksi.

5. YHTEENVETO

Työn aiheen tarjosi Wapice Oy. Wapice on suomalainen sulautettuja järjestelmiä tekevä yritys. Yksi Wapicen tuotteista on esineiden internetin datan keräämisen, visualisoinnin ja jälkikäsittelyn mahdollistava pilvipalvelu IoT-Ticket. Työn aiheena oli tutkia, miten IoT-Ticketiä voitaisiin laajentaa keräämään dataa myös uudesta NB-IoT-verkosta. Työssä tutustutuaan NB-IoT-verkkoon tekemällä, rakentamalla ja ohjelmoimalla sitä käyttävä demonstraatiolaite. Demonstraatiolaitteelle asetetut olennaisimmat vaatimukset ja ominaisuudet on esitetty Taulukossa 3.

Taulukko 3. *Lopullisen laitteen olennaisimmat ominaisuudet.*

Kuvaus	**Selite**
Työn tavoite	Laite, joka välittää anturidataa IoT-Ticketiin
Verkkotekniikka	Narrow Band-Internet of Things (NB-IoT)
Modeemi	Quectel BG96
Kehityslaitealusta	STM Nucleo L476RG
Tiedonsiirtoprotokolla	Constrained Application Protocol (CoAP)
Laitteenhallintaprotokolla	Light Weight Machine to Machine (LwM2M)
Anturit	Laitteen läheisyydessä olevat Bluetooth Low Energy (BLE) -majakat
Kehitysympäristö	Arm Mbed
Ohjelmointikielet	C++ (Sulautettu laite)
	Java, AngularJS (LwM2M palvelin)
Valmiskirjastot	Mbed client, Mbed Ble, Leshan

NB-IoT on eräs LPWAN-tekniikoista, joita ovat myös LoRaWAN ja Sigfox. LPWAN-tekniikoille yhteistä on, että niillä saavutetaan laaja kantama tukiaseman ja päätelaitteen välillä. Muita yhteisiä ominaisuuksia ovat pieni energiankulutus, matala päätelaitteiden yksikköhinta, suuri skaalautuvuus ja hidas ja harvoin tapahtuva datan siirto. NB-IoT:n erottaa LoRaWAN- ja Sigfox-verkoista se, että käytettävä taajuuskaista on lisensoitu. Verkko on myös pitkälti tehty uudelleenkäyttämällä LTE-verkon komponentteja. Käytännön etuna LTE-komponeneteista saadaan esimerkiksi taajuuskanavointi ja mahdollisuus priorisoida viestiliikennettä, mikä parantaa palvelun laatua. Etuna on myös suurempi mahdollinen datansiirtonopeus. NB-IoT:n heikkouksia muihin LPWAN-verkkoihin nähden ovat kalliimpi hinta ja huonompi energiatehokkuus.

Työssä käytetään datansiirtoon LwM2M-laitteenhallintaprotokollaa. LwM2M perustuu puolestaan kevyeeseen CoAP-tiedonsiirtoprotokollaan, jossa viestien pituus ja muut ominaisuudet on pyritty optimoimaan sovelluskohteisiin, joissa rajoittuneet laitteet pystyvät lähettämään yksinkertaista dataa palvelimelle mahdollisimman pienellä energiankulutuksella. LwM2M-protokollassa laite toimii palvelimelle kirjautumisen

jälkeen ikään kuin palvelimena, palvelin puolestaan kuin asiakkaana. Asiakkaassa olevat resurssit on jäjestetty loogisesti resursseihin, joita palvelin voi lukea, kirjoittaa tai suorittaa. Palvelin voi myös lähettää resurssiin tilauspyynnön, jonka jälkeen se vastaanottaa ilmoituksia resurssin arvon muutoksista ja näin ollen pystyy keräämään anturidataa säännöllisesti yhdellä kyselyllä. Resursseihin kirjoittaminen mahdollistaa laitteen etäkonfiguroinnin.

Työn NB-IoT-laite toimii yhteyspisteenä BLE-majakoille, joilta lähtöisin oleva anturidata lähetetään NB-IoT-verkossa eteenpäin. Järjestelmä osaa parsia anturidatan Googlen Eddystone -formaatista, Ruuvitag-majakoiden omasta raakaformaatista, sekä Cyalkit-majakoiden käyttämästä iBeacon-formaatin versiosta. Kaikissa formaateissa BLE-datan välitys on mahdollista ilman parinmuodostusta BLE-keskus- ja oheislaitteen välille. Sen sijaan datan siirrossa käytetään pelkästään BLE-mainosviestejä, joihin mahtuu mukaan noin 20 tavua hyötykuormaa. Kaikista MAC-osoitteista lähetetään yhteyspisteen kautta palvelimelle yksi viesti, jonka jälkeen palvelin voi määrittää tai olla määrittämättä MAC-osoitteen kiinnostavaksi. Kiinnostavista MAC-osoitteista lähetetään myös myöhemmät viestit edelleen palvelimelle.

Järjestelmän NB-IoT-laite toteutettiin ARM Mbed-OS -kehitysympäristön pohjalta. Kehitysympäristö tarjoaa reaaliaikakäyttöjärjestelmän, suuren määrän ohjelmistokirjastoja, ja on konfiguroitavissa yhteensopivaksi monien yhteistyökumppanien valmistamien kehitysalustojen kanssa. Laitteen käytössä olennaisin käytetty C++-kirjasto oli *Mbed client*, jolla voidaan toteuttaa LwM2M-protokollan asiakasrajapinta ja jäsentää kerättävä data loogisiin kokonaisuuksiin. Toinen tärkeä kirjasto oli *Mbed BLE*, jolla voidaan toteuttaa laitteen Bluetooth-rajapinta datan keräämiseksi BLE-majakoilta. Järjetelmän LwM2M-palvelin puolestaan toteutettiin käyttämällä *Eclipse Leshan* -projektin tarjoamia Java-kirjastoja.

LÄHTEET

[1] C. Anton-Haro, M. Dohler, Machine-to-machine (M2M) Communications: Architecture, Performance and Applications, nide 69, Woodhead Publishing Ltd, London, 2015;2014;.

[2] Arm mbed. Saatavissa: https://gitlab.exmachina.fr/fw-libs/mbed-os/tree/5.8.1/features/cellular/easy_cellular

[3] Arm mbed, Bluetooth low enegy -kirjastototeutus. Saatavissa: https://github.com/ARMmbed/ble/tree/master/ble

[4] Arm mbed, easy-connect ohjelmakirjasto. Saatavissa: https://github.com/ARMmbed/easy-connect

[5] Arm mbed, Nucleo pinout. Saatavissa: https://os.mbed.com/platforms/ST-Nucleo-L476RG/

[6] Arm mbed, Quectel BG96 modem driver. Saatavissa: https://gitlab.exmachina.fr/fw-libs/mbed-os/tree/5.8.1/features/cellular/framework/targets/QUECTEL/BG96

[7] L. Atzori, A. Iera, G. Morabito, The Internet of Things: A survey, Computer Networks, vsk. 54, nro 15, 2010, s. 2787–2805.

[8] Avnet-Silica. Saatavissa: https://os.mbed.com/teams/Avnet-Silica/code/Nucleo_NbIotBG96_A2_cloud_IBM/file/43647366c69e/main.cpp/

[9] T. Berners-Lee, L. Masinter, M. McCahill, Uniform Resource Locators (URL), 1994. Saatavissa: https://www.ietf.org/rfc/rfc1738.txt

[10] M. Centenaro, L. Vangelista, A. Zanella, M. Zorzi, Long-range communications in unlicensed bands: the rising stars in the IoT and smart city scenarios, IEEE Wireless Communications, vsk. 23, nro 5, October, 2016, s. 60–67.

[11] The Constrained Application Protocol (CoAP), Internet Engineering Task Force (IETF), Standard, kes. 2014. Saatavissa: https://tools.ietf.org/html/rfc7252

[12] Cypress, CYALKIT-E03 Solar-Powered BLE Sensor 5 Pack. Saatavissa: https://www.cypress.com/documentation/development-kitsboards/cyalkit-e03-solar-powered-ble-sensor-5-pack

[13] R. Davidson, Akiba, C. C., T. K., Getting Started with Bluetooth Low Energy, O'Reilly Media, Inc, 2014. Saatavissa: https://www.oreilly.com/library/view/getting-started-with/9781491900550/

[14] Eclipse, OMA Lightweight M2M server and client in Java. Saatavissa: https://www.eclipse.org/leshan/

[15] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, T. Berners-Lee, Hypertext Transfer Protocol, 1999. Saatavissa: https://www.ietf.org/rfc/rfc2616.txt

[16] N. Gupta, Inside Bluetooth Low Energy, Second Edition, second;second; ed., Artech House Inc, Norwood, 2016.

[17] M. Koühne, J. Sieck, Location-Based Services with iBeacon Technology, teoksessa: 2014 2nd International Conference on Artificial Intelligence, Modelling and Simulation, Nov, 2014, s. 315–321.

[18] Lightweight Machine to Machine Technical Specification, Open Mobile Alliance, Standard, hel. 2017. Saatavissa: http://www.openmobilealliance.org/release/LightweightM2M/V1_0-20170208-A/OMA-TS-LightweightM2M-V1_0-20170208-A.pdf

[19] A. Mbed, Getting started with mbed Client on mbed OS. Saatavissa: https://os.mbed.com/teams/mbed-os-examples/code/mbed-os-example-client/file/4ec747895c33/main.cpp/

[20] A. Mbed, IoT Device Development. Saatavissa: https://www.mbed.com/en/

[21] A. mbed, NUCLEO-L476RG. Saatavissa (viitattu 5.5.2019): https://os.mbed.com/platforms/ST-Nucleo-L476RG/

[22] K. Mekki, E. Bajic, F. Chaxel, F. Meyer, Overview of Cellular LPWAN Technologies for IoT Deployment: Sigfox, LoRaWAN, and NB-IoT, teoksessa: 2018 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops), March, 2018, s. 197–202.

[23] S. Microelectronics, X-NUCLEO-IDB05A1. Saatavissa (viitattu 5.5.2019): https://www.st.com/en/ecosystems/x-nucleo-idb05a1.html

[24] M.C. Ocak, Implementation of an Internet of Things Device Management Interface, mastersthesis, 2014. Saatavissa: https://aaltodoc.aalto.fi/bitstream/handle/123456789/14829/master_Ocak_Mert_2014.pdf?sequence=1&isAllowed=y

[25] R. V. M. Pereira, F. R. d. Sousa, E. A. Bezerra, M. D. Berejuck, A Digital Implementation of Eddystone Standard Using IBM 180nm Cell Library, teoksessa:

2017 VII Brazilian Symposium on Computing Systems Engineering (SBESC), Nov, 2017, s. 161–166.

[26] Quectel, LTE BG96 Cat.M1/NB1 EGPRS Module. Saatavissa (viitattu 5.5.2019): https://www.quectel.com/product/bg96.htm

[27] Y. Rama, M. A. Özpmar, A Comparison of Long-Range Licensed and Unlicensed LPWAN Technologies According to Their Geolocation Services and Commercial Opportunities, teoksessa: 2018 18th Mediterranean Microwave Symposium (MMS), Oct, 2018, s. 398–403.

[28] S. Rao, D. Chendanda, C. Deshpande, V. Lakkundi, Implementing LWM2M in constrained IoT devices, teoksessa: 2015 IEEE Conference on Wireless Sensors (ICWiSe), Aug, 2015, s. 52–57.

[29] Ruuvi, Sensor Protocol for Eddystone-URL. Saatavissa: https://github.com/ruuvi/ruuvi-sensor-protocols

[30] Ruuvi, What is Ruuvitag. Saatavissa (viitattu 5.5.2019): https://ruuvi.com/ruuvitag-specs/

[31] J. Sandoval, RESTful Java Web Services : Master Core REST Concepts and Create RESTful Web Services in Java, Packt Publishing, Olton, 2009.

[32] A. Silica, Getting started mbed NbIot with BG96 module. Saatavissa: https://os.mbed.com/teams/Avnet-Silica/code/Nucleo_NbIotBG96_A2_cloud_IBM/

[33] Solar-Powered BLE Sensor Beacon Reference Design Kit Guide, Cypress Semiconductor. Rev. D.

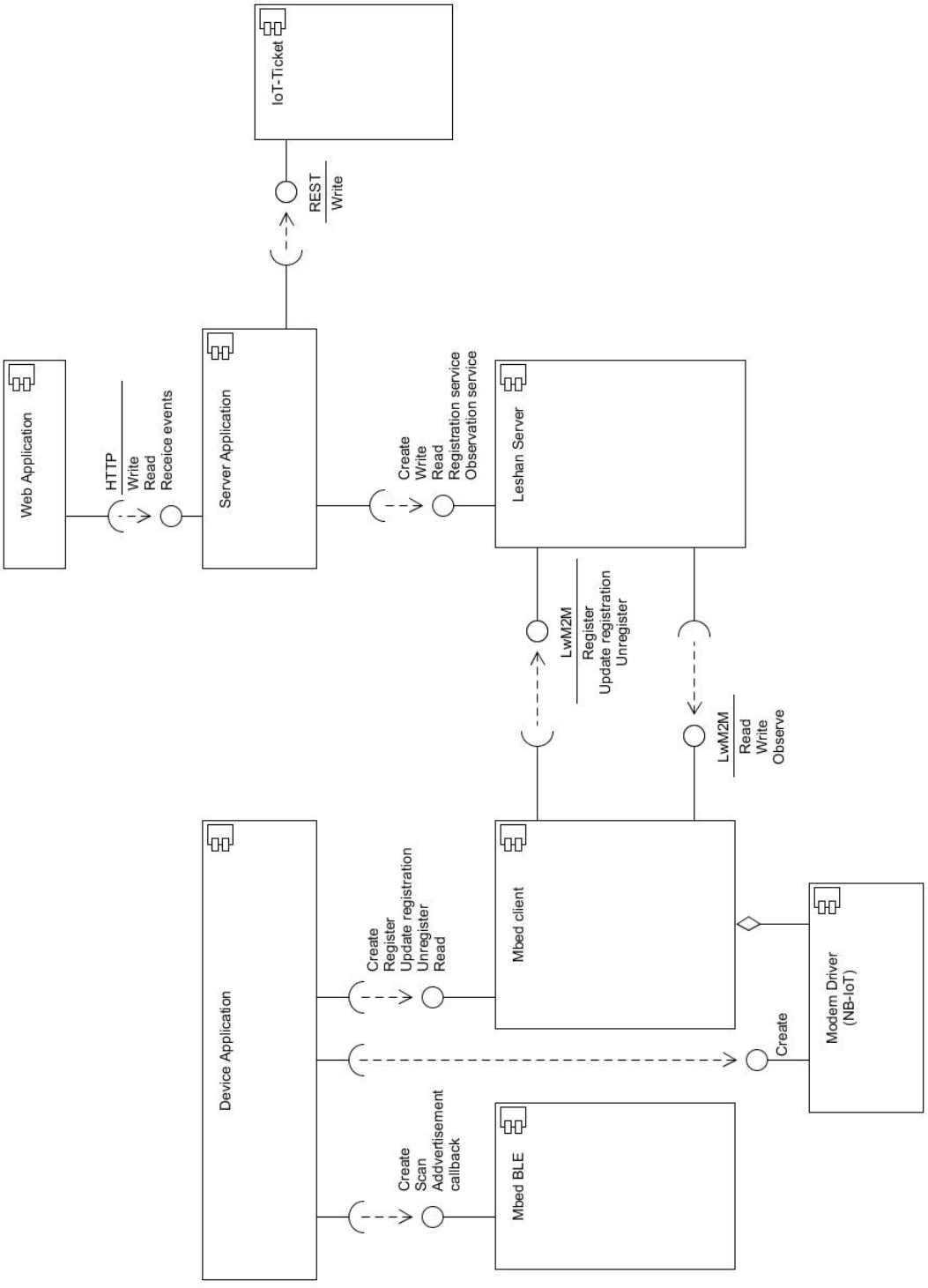
[34] Ublox, SARA-N2 series. Saatavissa (viitattu 5.5.2019): https://www.u-blox.com/en/product/sara-n2-series

[35] Y. P. E. Wang, X. Lin, A. Adhikary, A. Grovlen, Y. Sui, Y. Blankenship, J. Barbman, H. S. Razaghi, A Primer on 3GPP Narrowband Internet of Things, IEEE Communications Magazine, vsk. 55, nro 3, 2017.

[36] Wapice, IoT JavaClient. Saatavissa: https://github.com/IoT-Ticket/IoT-JavaClient

[37] S. Ziegler, Considerations on IPv6 scalability for the Internet of Things — Towards an intergalactic Internet, teoksessa: 2017 Global Internet of Things Summit (GIoTS), June, 2017, s. 1–4.

LIITE A: OHJELMISTON ARKKITEHTUURI



Kuva 33. *Ohjelmiston arkkitehtuurikuvaus*